Development of a New Uniform File Format for Neuroscience Data Across the Globe

Development of a New Uniform File Format for Neuroscience Data across the Globe

Claudia Friedsam

A Thesis in the Field of Biotechnology

for the Degree of Master of Liberal Arts in Extension Studies

Harvard University

May 2016

Abstract

With about 10^{11} neurons and about 10^{15} synaptic connections the human brain is among the most complex biologic systems ever observed (Rautenberg, Sobolev *et al*. 2011). Technical progress is driving the volume of data collection to unknown heights. Unsurprisingly neuroscience is rapidly increasing its already extreme volumes of collected data (Schwarz, Lebedev *et al.* 2014). At the same time the discipline is lagging behind with tools and concepts needed to fully exploit the information contained in the overwhelming amount of data. Scientific progress depends more and more on collaboration, which in turn requires data sharing (Teeters, Godfrey *et al.,* 2015). Neurophysiologic experiments are complex and diverse and the data are recorded in a multitude of different data formats. This lack of compatibility poses a major obstacle to efficient data sharing and collaboration (Breaking Down the Data Barriers in Neuroscience 2014).

Neurodata without Borders (NWB) is a new initiative, which aims at overcoming these limitations by creating a new common file format, the NWB format, to facilitate data sharing across the individual labs. The objective of this thesis is to investigate how well this newly created file format is suited to become the new common standard within the research community of cell-based neurophysiology. To address this question we studied how well the NWB file format encodes the diversity of existing data in this field. Four carefully selected datasets served as model cases for the development and testing of the new file format. We investigated how diverse these datasets are and how well they

represent the different data categories that the workgroup defined as essential for the field of cell-based neurophysiology. Furthermore, we determined the degree of standardization that the new file format achieves by studying the amount of empty fields and special custom fields needed to map the four datasets onto the NWB format. To further benchmark the NWB format we compared to existing file formats and studied if and how NWB overcomes the limitation of the other formats. Finally, we developed a metric to evaluate how well the NWB format fulfills the requirements for the new common file format, which were defined by the NWB workgroup. We hypothesize that the NWB format is suited to become the common file standard within the community of cell-based neurophysiology.

The results of this study demonstrate how much the NWB file format has already achieved. However, due to the small size of test datasets and the resulting lack of diversity we reject our hypotheses at this point in time. More results are required to get clarity about the true potential of the NWB format and time will show if it will rise to become the universal data standard in the field of cell-based electrophysiology.

Acknowledgements

First I would like to thank my thesis director Barry Wark. This work would not have been possible without his ongoing support, efforts and advice.

I would also like to thank Dr Steven Denkin for his guidance while making the research proposal for this topic acceptable and for his help with reviewing and editing this thesis.

Finally I would like to thank the entire NWB work group for all their efforts and the materials provided to get this thesis underway.

Table of Contents

List of Tables

List of Figures

Chapter I

Introduction

The following chapter provides some background information about file formats for neurophysiological data in general and the Neurodata without Borders (NWB) project in particular. It furthermore outlines the research question and the methods applied to address it.

Motivation and Need for a Uniform Data Format

Neurophysiology is an extremely dynamic and fast moving discipline. It has undergone a rapid development since the introduction of the single unit voltage-clamp by Huxley and Hodgkin (Hodgkin and Huxley 1952). Recent technical advances allow neuroscientists to generate data that is substantially different from these early data sets. Today, researchers can simultaneously observe the activity of a thousand neurons in an animal's brain while the animal is performing a specific task (Schwarz, Lebedev *et al.* 2014). These numbers are expected to reach the millions in the near future. Researchers across the globe today are collecting a huge amount of data that is in principle compatible. However, they use different formats and methods to record, document and evaluate these data. This lack of compatibility makes it challenging to share and exchange experimental data and analysis methods and to maximize their impact (Breaking Down the Data Barriers in Neuroscience 2014).

Thanks to standardized file formats, it is easy to share files like pictures, documents or video. Standardized software tools interpret these data files and make the

information available for the user. These are standard procedures that most people perform multiple times per day without feeling any particular difficulty. The situation is more complicated when it comes to sharing data files and results in a life science like neurophysiology. Neurophysiology experiments are in essence complex and diverse. They are recorded from various species, with a variety of electrode configurations multitude of different types of signals and many levels of study (Sobolev, Stoewer *et al.* 2014). Data is stored in different, typically ad-hoc, formats and it is described in a multitude of ways, as no common standard file format exists (A Standard for Neuroscience Data 2015). To add up to the problem neuroscientists use a wide variety of software to acquire, analyze and visualize electrophysiological signals (Garcia, Guarino, *et al.* 2014). Recent advancements in technology and methodology during the past years have furthermore increased both the volume and complexity of data recorded in electrophysiological experiments (Sobolev, Stoewer *et al.* 2014). One consequence of this increase in complexity is the need for a sophisticated method of documentation (Zehl, Denker *et al.* 2014). Experimental metadata typically have to be collected from various sources and in different formats (e.g different measurement devices, handwritten notes, etc.) (Sobolev, Stoewer *et al.* 2014). Often, each lab defines its own methods, procedures, and format conventions for organizing and managing this information. These 'metadata' are necessary to reproduce the study but are also essential for searching, selecting, and analyzing the data (Grewe, Wachtler *et al.* 2010). They are only rarely available in a structured, comprehensive, and machine-readable form (Grewe, Wachtler *et al.* 2011). Thus, for a multitude of reasons, data sharing often requires substantial effort to make the data useable in one form or another (Teeters, Benda *et al.* 2013).

At the same time data sharing is of the utmost importance in the field of neuroscience. Scientific progress depends increasingly on collaboration, which implies exchange of data and re-analysis of previously recorded data (Rautenberg, Sobolev *et al.* 2011). Many unresolved aspects of brain function could only be addressed by a combination of recent technologies (Teeters, Benda *et al.* 2013). A culture of data sharing is a fundamental ingredient to successfully address these challenges. The need for a common data format is further increased by the BRAIN (Brain Research through Advancing Innovative Neurotechnologies) initiative started in 2013 by President Barack Obama (A Standard for Neuroscience Data 2015). This initiative was started to "revolutionize the understanding of the human mind" and to address brain diseases like e.g. Alzheimer's or autism (BRAIN Initiative 2015). The BRAIN initiative is meant to be a counterpart to the Human Genome Project with the goal to promote the development and application of new technology to explore major brain function and elucidate the complex links between brain function and behavior (BRAIN Initiative 2015). To achieve these goals it is also expected that the initiative will generate a "deluge of new data" comparable to the output of "the 17-mile-long Hadron Collider" (A Standard for Neuroscience Data 2015).

This challenge is further increased by the fact that the European HBI ("Human Brain Initiative") and the US BRAIN initiative have announced to launch a collaboration (Reardon 2014). To handle the expected massive amounts of information from these collaborative efforts it is mandatory to develop the required data-sharing infrastructure (A Standard for Neuroscience Data 2015). Developing a common file format for neurophysiology is a tedious and difficult task, but it is more important than ever to successfully manage the upcoming challenges.

Past Initiatives and File Formats

A multitude of past efforts have been undertaken to facilitate data sharing in neuroscience. Many different file formats have been developed. One of the earliest candidates goes back to 1992, when Kemp et al. made a first attempt to create a common file format for polygraphic recordings in the subdomain of clinical electroencephalography (EEG) (Kemp, Värri *et al.* 1992). This file format was originally used by 7 laboratories and was later on succeeded by EDF ("European Data Format")(Kemp and Olivan 2003) and GDF ("General Data Format" for Bioscience)(Schlögl 2006) and had some success in its domain. Numerous other data formats for neuroscience have been developed for a multitude of different reasons. Schloegl (2010) compares 19 different commonly used biomedical data formats. Other prominent examples are the KWIK format (Klusta-team/kwiklib 2014), the LBNL Brain format (Brainformat 2015) the MEF format (Multiscale Electrophysiology File Format 2014) and the NIX format (G-Node/nix 2014). The KlustaKwik spike-sorting suite, an open-source spike sorting software for multielectrode extracellular recordings, uses the KWIK format. It is based on HDF5 and is optimized for high-channel count electrophysiology data. The LBNL Brain format is another HDF5 based format. It is based on the concept of managed objects and is able to handle data of different types. The Multiscale Electrophysiology File (MEF) format is not based on HDF5. It is mainly used for EEG data. The Nix format is a C++ based data format with Python wrappers. It is based on a generic data model for neuroscience and is HDF5 based.

Alternative to developing a common file format, it is also a possibility to define an application-programming interface (API) to handle data conversion. This is provided as a

library, which can be used to facilitate uniform access to a multitude of different data in different file formats. Prominent examples for this approach are the Neuroshare API and the NEO API (Neo – NeuralEnsemble 2015 and Garcia, Guarino *et al*. 2014).

The Neuroshare API reads data from different file formats provided by hardware manufacturers. It defines four basic data types that have been included in many subsequently developed systems including Neo. The Python objects defined in Neo can be used to represent electrophysiology data and interface to different backends e.g. Neuroshare or HDF5.

All these approaches represent a unique solution, solve some specific problems and have their own user groups. None of them however has become a common standard for neuroscience.

There have also been attempts to create a common standard for metadata in neuroscience, which are essential for data sharing and should be part of any successful common file format. In 2009 a set of minimal common metadata (MINI) has been proposed by Gibson *et.al.* (Gibson, Overton *et al.* 2009). Grewe *et al.* describe a "bottom-up approach" to develop a format for data annotation called open metaData Markup Language" (odML) (Grewe and Wachtler 2011). This format for metadata transfer is not based on a specific, complex metadata model. It has some flexibiliy while it still can be used in a standardized way to ensure interoperability and thus achieves some level of standardization for metadata. Recently some efforts have been started to develop 'OEN', "Ontology for Experimental Neurophysiology" (Le Franc, Bandrowski *et al.* 2014), which extends odML and other existing ontologies.

Numerous initiatives have been started to support data sharing. Many of them

provide a space for common data and engage in initiatives which aim e.g. at developing a file format, developing data management techniques, a common ontology etc.

One of the most important examples is the International Neuroinformatics Coordinating Facility (INCF). This is an international science organization, which aims at supporting cooperation in the neuroinformatics field. It was founded in 2005 by recommendations of the Global Science Forum working group of the OECD, with initially sixteen participating countries (International Neuroinformatics Coordinating Facility, 2015). But its roots go back to 1998, when the recommendation to coordinate international efforts was documented in the report on Bioinformatics in connection with the OECD Mega science Forum (International Neuroinformatics Coordinating Facility 2015). Today the INCF has 17 members: Belgium, the Czech Republic, Finland, France, Germany, India, Italy, Japan, the Netherlands, Norway, Poland, the Republic of Korea, Sweden, Switzerland, the United Kingdom, the United States and Victoria, Australia. Each country has a National Node that coordinates national neuroinformatics activity and is connected to the Secretary, which is located in Stockholm, Sweden (INCF Neuroinformatics Portal 2015). The INCF currently has four program areas: “Digital Brain Atlasing”, “Ontologies for Neural Structures”, “Multi-scale modeling” and “Standards for Data Sharing” (INCF Neuroinformatics Portal 2015). Furthermore it provides the INCF Dataspace, a cloud based file system to host different kinds of neuroscience data (INCF Neuroinformatics Portal 2015). The INCF “Standards for Data Sharing” aim at creating standards and tools for data sharing. It currently focuses on two areas, namely neuroimaging and electrophysiology (INCF Neuroinformatics Portal 2015). Each country makes its own contribution to any of the four different areas.

Examples are the NIX file format, the Neo API, the Neuroshare API and odML, which have been developed by the German Node (G-Node) of the INCF (Welcome to the German Neuroinformatics 2015).

The neurodatabase.org project was one of the first public databases developed by Gardner in 2004 (Neurodatabase 2015). It makes use of an elaborate data model and also provides a query protocol to facilitate the exchange of neurophysiological data (Gardner 2004). The data are provided in a custom format and contain extensive metadata (Gardner 2004).

CARMEN is another project that was started in 2006 and went into a second phase in 2010 (Welcome to CARMEN 2015). It provides a database for both private storage and public data exchange. Furthermore CARMEN has developed its own file format Carmen NDF for neuroscientific data, which stores electrophysiology data in MATLAB file format and provides some procedures for data analysis (Fletcher and Liang 2008). Metadata are optional and can be entered by the user following the standard of MINI as described by the CARMEN consortium (Gibson, Overton *et al.* 2009).

The Collaborative Research in Computational Neuroscience is a joint initiative of the National Institute of Health (NIH) and the National Science Foundation (NSF), which has been started in 2002 (Welcome to the CRCNS data sharing website 2015). It supports the integration of theoretical and experimental neuroscience through collaboration and provides publically available data from various sources (Welcome to the CRCNS data sharing website 2015). Since recently it also focuses its efforts on various aspects of sharing data and tools (Welcome to the CRCNS data sharing website 2015).

All these examples for common file formats and data sharing platforms and tools have

had some success in specific domains, but so far none of them has been accepted as a global standard in the field of neuroscience. Nevertheless they are an important resource for the development of the new file format.

The NWB Initiative

This section provides some background information about the goals and the progress of the NWB initiative.

Goals of NWB

Neurodata Without Borders (NWB) is a recent initiative that aims at standardizing neuroscience data on an international scale (Recent News 2014). It was initiated by the Kavli Foundation as a direct response to the BRAIN Initiative and the planned public release of many larger neurophysiology datasets by the Allen Institute for Brain Science (AIBS) in Seattle (Recent News 2014). Kavli, General Electric, Janelia Farm, the AIBS and the INCF support the NWB project. Currently there are a handful of collaborating laboratories that provide datasets and resources, namely the Buzsaki group at NYU, the Svoboda group at Janelia Farm, the Meister group at Caltech and the AIBS in Seattle (Science Funders Hope to Link Up Large Scale Brain Research 2015). NWB is a one-year project with the goal to standardize neuroscience data on a global scale to facilitate data sharing by researchers worldwide. The first project is called "Neurodata Without Borders: Neurophysiology" (Recent News 2014). The initial one-year program, which started in August 2014, focuses explicitly on cell-based neurophysiology data, an area with extensive interactions between experimentalist and theorists (Recent News 2014). This new attempt is similar in many aspects to the past ones. However, it is happening on

a larger scale, with some carefully selected high quality datasets, which are well documented (Recent News 2014). The data should catch the interest of the scientific community and be easy to navigate in the final format, which is not straightforward to achieve. Furthermore the partners plan to set an example and migrate the data of their groups to the new data format to create some momentum in the community (Recent News 2014). Along the road the initiative is calling on the neuroscience community for input, with the hope that scientist provide ideas and maybe their own data format as a candidate (Recent News 2014). The team also plans to work directly with external software developers and vendors at various stages of the project (Recent News 2014).

The new format has to be able to address all the specific requirements to store electrical and optical recordings of various types and include complex metadata to make the data useable for other scientists (Recent News 2014). At the same time it has to be flexible and extensible. API's will be developed to make the new data format usable for the individual labs and third party software used in the community (Recent News 2014). The timeline of one year is rather ambitious, so the expectation is not to get everything "100% right for 100% of all researchers" (Recent News 2014), but to get up to a successful start to face the upcoming challenges in the neuroscience fields. The investigations suggested for this project are designed to support the development process and study the prospects of success during all stages of development, which has not been done in all the past approaches.

Phases and Milestones for the NWB project

The timeline for the NWB project is illustrated in Figure 1 in Appendix I. The project is divided in three phases with two Hackathons and two important development milestones.

Phase 1: months 1-4

The project started in August 2014. In the beginning the workgroup focused on analyzing the model datasets and the associated use cases. They worked out general requirements for a common file format and criteria considered for a metric to assess existing file formats. This was in direct preparation for Hackathon 1.

Hackathon 1:

Hackathon 1 took place November 20-22, 2014. It focused on presenting and evaluation existing data formats and getting input from the community. The goal was to select suitable candidates and defining the requirements for a common file format.

Phase 2: months 5-8

After the Hackathon the group developed the first version of the common data format. It was based on the Allen Institute 'Orca' format and integrated ideas from other formats. The group converted the datasets to this new format by the means of a python API. Some example applications were develop to test the functionality of the data files in the new format and feedback from the different data providers was evaluated, to refine the file format before Hackathon 2.

Hackathon 2:

Hackathon 2 took place May 15-17, 2015. The main focus was to collect feedback from the users and to identify the most important changes to focus on during the last phase of

development. The major conclusions were that the write API for the new format should utilize a high-level specification language to make it more user-friendly and that the community needed a native MATLAB based write API.

Phase 3: months 9-12

In the last phase of NWB the group finalized both the specification language based python and the MATLAB APIs with the required documentation. The API's were released in the first week of August 2015. Since then the project is open source and the community is exploring the new file format.

Research Question

The proposed research project is part of the NWB initiative. The initial one-year effort focused on a small subset of representative data in the area of cell-based neurophysiology. The new data format stores electrical and optical measurement data of neurological activity as well as descriptive metadata, which aim to document important information like the experimental conditions or the animal behavior.

One of the big challenges is that the new file format has to accommodate all the distinctive parts of an individual dataset to be useful while at the same time represent a generic standard. The participating labs provide a total of five datasets accessible at the webpage of "Collaborative Research in Computational Neuroscience" (CRCNS, crcns.org/datasets). These datasets and the experiments that generated them were used as model data to define a common standard for electrophysiological and optophysiological data. The outcome of this process was used as the basis to define the new file format. The functionality of the new file format was reviewed by the participating labs and tested based on use-cases for the original data sets. This ensured that the labs could perform the

same types of analysis with data in the new format while producing the same outcomes as with the data in its ad-hoc format.

The hypothesis of this research project is that this newly created file format is suited to become the common file standard within the community of cell-based neurophysiology. This is supported by the fact that the new file format

- is able to represent the diversity of existing data in this field and is flexible enough to accommodate the output of future experiments in this area
- shows a sufficient degree of standardization
- overcomes weaknesses of existing file formats
- fulfills the requirements for the new common file format as defined by the NWB workgroup

We studied the dimensions of usefulness and standardization by thoroughly evaluating the information provided for the original datasets and by studying the mapping of the original datasets onto the new file format. For a particular dataset it is especially important to determine the number of extra fields needed to represent specific parts of the data set in the new format and the number of empty fields that are introduced by the new file format but unnecessary for the given dataset. These two proportions directly relate to the degree of standardization that the new file format can achieve. Adding only a small number of additional fields in the new file format to cover diverse data sets and a majority of populated pre-defined fields in the new file format are good indicators of standardization.

To judge the information gained from this investigation we also determined the degree of diversity of the original data files. This is important with respect to

standardization, as the results gained will strongly depend on this parameter. It is furthermore important to determine if the new file format does indeed accommodate the wealth of existing types of data files in the field.

Furthermore, we compared the new file format to existing ones by studying the different data models. We specifically investigated how the different file formats address key requirements for the common file format as identified by the NWB workgroup and compare this to the solutions that the NWB file format provides. This allowed identifying potential strengths and weaknesses of the new file format as compared to others.

There are more aspects that are likely to determine the future success, which were defined by the NWB workgroup. During the course of this project we developed a metric to capture the most important determinants for success and applied it to the file format in its stages of development. The metric has different dimensions, some inspired by engineering questions, others by the specific requirements of the users involved in the development process. The different aspects defined a quantitative measure to describe the performance of the new file format and the progress made during the development phase.

The results of our studies serve to evaluate our hypotheses. Independent from that they also provide insights into the determinants for success or failure to address the important task of creating a common file format in the field of neuroscience. A common file format is the basis for successful data sharing and collaboration. Facing the huge amount of data made publicly available like, e.g. with the newest data release of the Allen Institute for Brain Science (Neurodata Without Borders meeting report, 2015), solving this problem seems more important than ever.

Chapter II

Materials and Methods

This chapter introduces the model data sets, describes existing file formats and the process of creating the new NWB file format. It also explains the methods used to address the research question.

Model Data Sets

Currently five data sets are used as model cases for the new file format. They are all available at CRCNS (Data Sets 2015). An overview is given in Table 1 in Appendix II.

In phase 1 of the project the datasets have been described in the form of entity relationship diagrams, use cases and data analysis goals. The results of this analysis have been summarized for the first Hackathon (NWB_hackathon1.pdf 2015) and can be found in Appendix IV. CRCNS pvc-6 is the only dataset, which has not been used for testing during the earlier stages of development and will thus be left out in this research project. The other four datasets are described below in more detail.

CRCNS hc-3: Rat Hippocampus, Gyorgy Buszaki, New York University

The CRCNS hc-3 data set contains multiple single unit recordings from different rat hippocampal and entorhinal regions while the animals were performing one of 14 behavioral tasks. It is a large collection of data with 7736 cells recorded from 11 animals

(crcns-hc3-data-description.pdf, 2015). Experimental details and conclusion are explained in Diba, K., Buzsáki, G. 2008 and Mizuseki, Sirota, *et al* 2009.

The data files are organized in sessions that belong to a top-level directory. The top-level directories contain data for sessions recorded on the same day with the same animal and the same electrode placement combination. All of the data in a specific top-level directory are concatenated for spike sorting (crcns-hc3-data-description.pdf, 2015). The session data contain wide-band raw data, post-experiment processed LFP (local field potential) data, spike times, spike waveforms, features for PCA, and clustering results (crcns-hc3-processing-flowchart.pdf, 2015).

The metadata are organized in 3 different zip files: crcns-hc3-channelorder.zip, crcns-hc3-metadata-tables.zip and crcns-hc3-original-docs.zip. They contain information about the relative depth in the brain of different parts of the recording electrodes for many of the sessions, the cells, brain regions, experiment sessions, files in the data set, etc. (crcns-hc3-data-description.pdf, 2015)

The data are stored in a collection of binary and ascii files with different endings. The data contain:

1. raw data (e.g. ec*.dat)
2. LFP (local field potential) data (ec*.eeg)
3. time of spike (ec*.res)
4. spike waveform (ec*.spk)
5. PCA features used to do spike sorting (ec*.fet)
6. class (classification) for each spike, i.e. result of spike sorting (ec*.clu)

Figure 2 in Appendix I gives a comprehensive overview over all the files associated with

the data. Figure 3 in Appendix I shows how processed data are derived from the raw data.

CRCNS alm-1: Electrophysiological Recordings from Rat Barrel Cortex, Karel Svoboda, Janelia Farm

The experiments of CRCNS alm-1 explore neural dynamics in the mouse anterior motor cortex (ALM) and its relationship to voluntary movement (about-alm-1, 2015). The datasets are from 19 animals and 99 recording sessions (crcns_alm-1_data_description.pdf, 2015). The data consist of extracellular recordings from ALM neurons in adult mice using 32 channels (about-alm-1, 2015). The mice use whiskers to detect a pole position and indicate their decision by licking on the left or on the right (crcns_alm-1_data_description.pdf, 2015). Antidromic spikes were induced by photostimulation in either layer 5 intratelencephalic neuron axons through a cranial window over contra-lateral ALM or pyramidal tract neuron axons via an optical fiber in ipsi-lateral brainstem (about-alm-1, 2015). More details are given in Li, Chen, *et al.* 2015 and Guo, Li, *et al.* 2014.

The data consist of (about-alm-1, 2015):

- Raw extracellular recordings
- Processed data (spike sorted units)
- Photostimulation data
    - Types
    - Timings
- Behavior
    - Tasks

- Timings
- Animal response

The data for each session consists of three .mat files (crcns_alm-1_data_description.pdf, 2015):

- One file contains the raw voltage traces from each trial. The total size of the raw data is ~ 700 GB
- One file contains comprehensive metadata for the given session.
- The third file is the "data structure file", which contains the processed data. The compressed size of all data structure files is ~ 13 GB.

The more recent data upload contains 99 sessions. Apart from these there are three older sessions available. These older data are the ones that have been used for this research project (about-alm-1, 2015).

The structure of the data structure file is shown in Figure 4 in Appendix I. A detailed explanation of the data entries is given in Figure 5 in Appendix I (adapted from crcns_alm-1_data_description.pdf, 2015).

CRCNS ssc-1: Calcium imaging of rat somatosensory cortex, Karel Svoboda, Janelia Farm

The CRCNS ssc-1 datasets contain calcium-imaging data from vibrissal S1 in mice performing a pole localization task (about-ssc-1, 2015). Data from five sessions are provided. The mice performing in these experiments use a single whisker to indicate a pole position (about-ssc-1, 2015). After a delay, the mice suggest a pole position by either licking right or left. Images were acquired by the means of 2-photon imaging in

conjunction with a protein calcium sensor (crcns_ssc-1_data_description.pdf, 2015). The raw data consist of calcium imaging stacks and matching whisker videos. These raw data were used to determine fluorescence time series from multiple individual cells together with simultaneously recorded behavioral variables, e.g. whisker dynamics (crcns_ssc-1_data_description.pdf, 2015). More experimental details are described in Peron, Freeman, *et al.,* 2015.

The data contain (about-ssc-1, 2015):

- Raw data
    - Imaging stacks
    - Whisker videography
- Processed calcium data
    - Masks of cell locations (Regions of interest, or ROIs)
    - Raw fluorescence for each ROI
    - Change in fluorescence for each ROI
    - Algorithmically extracted calcium events for each ROI
- Processed whisker video
    - Angle of whisker
    - Curvature of whisker
    - Times at which whisker contacts the stimulus pole
- Behavioral data
    - Trial types
    - Trial timing
    - Trial outcomes

The raw data are very bulky (~ 154 GB in compressed form) and are stored separately. The calcium imaging stacks are stored as tif files with additional metadata stored in separate files (about-ssc-1, 2015). The raw whisker videos are provided as mp4s. Processed data are saved in matlab structures as .mat files. Each .mat file contains one or multiple sessions. The processed data provided at CRCNS only add up to ~600 MB in volume (about-ssc-1, 2015).

Figure 6 in Appendix I shows the file structure and indicates where the different data are contained. Figure 7 in Appendix I provides a detailed description of the data entries (adapted from crcns_ssc-1_data_description.pdf, 2015)

CRCNS ret-1: Retina, Markus Meister, Caltech

The data contain single-unit neural responses of retinal ganglion cells to various visual stimuli (crcns_ret-1_data_description.pdf, 2015). They were recorded from isolated retina from lab mice collected by using a 61-electrode array (crcns_ret-1_data_description.pdf, 2015).

Data are from 16 sessions with no raw data provided. The processed data (spike times) for all units in a given session along with information needed to generate the stimulus are provided in a .mat file per session. The total (compressed) size of the data is ~70 MB (about-ret-1, 2015). Two additional files are provided (crcns_ret-1_data_description.pdf, 2015):

- Ran1.m: a random number generator for visual stimuli
- Ran1.bin: 3e8 binary numbers generated by ran1.m with the seed -10000

A more detailed description can be found in Lefebvre, Zhang *et al.*, 2008.

The file structure of the .mat files is shown in Figure 8 in Appendix I.

The data file entries are explained in Figure 9 in Appendix I (extracted from (crcns_ret-1_data_description.pdf, 2015)).

File Formats

This section explains the general requirements for the new file format, discusses the main features of existing file formats and describes the design and development of the NWB file format.

General Requirements for the new File Format

During the first Hackathon (November 20-22, 2014 at Janelia Farm) the participants identified the following general requirements for the new file format (Neurodata Without Borders meeting report 2015):

- ability to represent many types of data collected in complex neurophysiological experiments, e.g. time series (like voltage traces), image stacks (optical imaging), stimuli, etc.
- extensibility: the file format should be able to support future experiments which are hard to predict. This could require new fields, new parameters or just challenge the data format in terms of dimensions and performance.
- usability: the file format should be user friendly and intuitive
- usefulness: the file format should be beneficial for research, it should facilitate data analysis in an efficient way and support the use of open source software tools and it should certainly support file sharing in an efficient way

- be adopted by a greater community: this among other factors depends on the previously listed criteria.

Apart from these very generic criteria the group identified some more specific characteristics of the file format (Personal notes, taken at the Hackathon):

- Provenance: this describes the ability to track derived data through each stage of the analysis.
- Standardization: keep the file format as compact as possible and prevent unnecessary extensions.
- Platform independence: make the file format useable for the most common different platforms (Windows, MAC OS, Linux distributions)
- Language support: the file format should support the most commonly used languages including Python, Matlab, Igor, Java, C++
- Performance criteria: e.g. high read bandwidth, file access time, processing time, storage requirements
- Basic requirements: The file format has to be open source and it is supposed to be HDF5 based

Furthermore, the group identified several problematic questions, which would have to be addressed at least in some preliminary way:

- The file format should not include bulky raw data, to limit the file size, however, the raw data should be at least somehow connected to the files and be accessible
- Different recording devices have different sampling rates. All entities within the file format should be consistently related to a common timeframe.
- Metadata pose a special challenge. While it is highly desirable to incorporate

standardized machine-readable metadata, this is not easily achieved. For many experiments there will be highly specialized metadata which can't be standardized but have to be included in the file to adequately describe the meaning of the data

In order to successfully establish the new file format all of these issues have to be addressed in some way. To achieve this, the group integrated many ideas from existing file formats. The most important ones are described in the following paragraph.

Existing File Formats

After Hackathon 1 the NWB work group members picked the Orca format as a basis for the new file format, which is described in the next section. Furthermore they focused on taking a closer look at four other existing file formats: the KWIK format, the Svoboda lab file format, the Nix format and the LBNL Brain format. Each of them is described in more detail below.

The Kwik Format. The 'Kwik' format is the underlying file format for KlustaSuite, a spike sorting software for multi-electrode extracellular recordings. It is a standardized, general-purpose data format based on HDF5. The API is written in python. The data are stored in three different files (Klusta-team/kwiklib, 2015):

- KWD file: raw/unfiltered data, typically tens of GBs
- KWX file: spike data, features, masks, waveforms, typically tens of GBs
- KWIK file: main file, contains:
    - metadata
    - spike times
    - clusters
    - recordings for the spike times

- probe-related information
- information about channels
- information about cluster groups,
- events and event-types
- aesthetic information, user data, information data

Figure 10 in Appendix I shows an excerpt of the structure of a KWIK file.

KWIK is user friendly and easy to read. Bulky raw data are strictly separated from the main file. Its modular design makes it easily extensible.

Svoboda lab file format. The Svoboda lab format is a Matlab-based format, which was designed to be flexible and extensible. In this format a dataset consists of (Svoboda_lab_data_format_general.pdf, 2015):

- raw data
- processed data
- metadata

Data are stored in a "session object", mostly in the form of hash tables. 'Heavy' raw data like e.g. images are not directly included in the session object but only referenced (Svoboda_lab_data_format_general.pdf, 2015). 'Light' raw data are included but not provided in raw format. The experiment description is included in the field metaDataHash and makes use of the vocabulary defined in the meta_data_template file (Svoboda_lab_data_format_general.pdf, 2015). Additional metadata are provided in other parts of the session object. However, extensive metadata are stored in a separate file. The file format can currently describe three different types of experiments (Svoboda_lab_data_format_general.pdf, 2015):

- single electrode electrophysiological (cell attached or intracellular)
- multi-electrode electrophysiological (silicon probes, tetrodes, etc)
- cellular calcium imaging data

The format makes extensive use of key-value pairs (hash tables) to facilitate searching and simplify code, which works with the session objects. The use of hash tables makes extensions easy and it also enforces field check by code that works on a session object as data can only be accessed by retrieving a value for a particular key (Svoboda_lab_data_format_general.pdf, 2015). The data format distinguishes time series with data points evenly spaced in time and event data with sparse timestamps. Accordingly the basic types are timeSeries and eventSeries with timeSeriesArray and eventSeriesArray to store the matching data (Svoboda_lab_data_format_general.pdf, 2015).

Two examples for a Svoboda lab file have been discussed in the data description section above.

Nix format. Nix utilizes HDF5 and is implemented in C++. It is based on a fixed data model with six elements (Block, DataArray, Tag, MultiTag, Source and Group) that is capable of storing different types of data (G-Node/nix, 2015). All data, along with units and metadata are stored in a Block. DataArray holds the data, whereas Tags and MultiTags can be used to define "Regions of Interest" (ROIs) within the data (G-Node/nix, 2015). Source describes the provenance of the data (G-Node/nix, 2015). Group defines subgroups below the block (G-Node/nix, 2015). The data model is shown in Figure 11 in Appendix I (G-Node/nix, 2015).

Metadata are stored by utilizing a model similar to the open metadata Markup Language (odML) model (Grewe , Wachtler *et al*., 2010). All major entities of the data model contain a metadata field, which can be used to link metadata to the specific entity (G-Node/nix, 2015). This is realized by pointing to the id, which marks the relevant metadata section (G-Node/nix, 2015).

Nix has a variety of advantages. The data model is very flexible; it basically works with any type of data. It also facilitates common tasks like linking hetereogeneous data with a common time base or labeling raw data with measurement units (Neurodata Without Borders meeting report 2015). Nix is fairly well developed. It has bindings for Python and Java and it comes with unit tests, tools (for browsing, validation and benchmarking) and extensive documentation.

LBNL Brain format. The LBNL BRAIN format is an HDF5 based format developed for Electrocortigraphy (ECoG) data. The format and its API are based on the concept of so-called “managed objects” (Brain Format, 2015). A managed object provides all functionality required to manage data of a given type. Managed objects provide (Brain Format, 2015):

- a formal specification of the object’s format
- validation of format compliance
- initialization of required elements
- easy access to data content
- a specification of primary datasets for vis and analysis

Managed objects can also be nested and/or extended for reuse and modular design. The API defines modules for raw ECoG data, processed ECoG data as well as reusable,

generic modules for annotations, which allow defining subsets of data (Brain Format, 2015).

The basic structure of a Brain format file is shown in Figure 12 in Appendix I (Brain Format, 2015).

The Brain format has a variety of advantages. It is user-friendly, includes a JSON specification and has options for file verification in place (Brain Format, 2015). Furthermore, this file format supports the possibility to define sub modules that could be used as building blocks for different types of experiments. It would allow for specialization as managed objects can be extended and support standardization at the same time as sub modules could be reused across labs (Neurodata Without Borders meeting report, 2015).

Development of the NWB File Format

Several candidates for the data format have been discussed before and during the first Hackathon and it was finally decided to use the 'Orca' file format, an internally used file format at the Allen institute as a starting point for the new file format.

This file format was modified to integrate the requirements defined in the 'what' document by the researchers as well as some main ideas of other file formats, namely the 'Kwik' file format, the 'Svoboda lab' file format and the 'Nix' format.

The interim outcome of this integration process was named 'Borg' format and was used for the first file conversions of the different data sets.

The individual labs provided some concrete feedback on the converted data files and the team made some minor adjustments to accommodate for these. The major issue however

was that the file format was too difficult to understand for the users and not quite flexible enough. This triggered the creation of a specification language: a JSON file that would basically describe the different parts of the file format and build a bridge to make it easier to understand and handle the file format.

By the end of the project the new specification language based write API was completed in both Python and Matlab and was released as an open source project.

The 'Orca' Format: the Ancestor of the NWB format. The Orca format was designed for internal use of the Allen Institute. Its design targets a wide variety of use-cases to account for the diversity of data at the institute to store intra- and extra-cellular electrophysiology data as well as 2-photon and intrinsic imaging opto-physiology data (NWBh1_09_Keith_Godfrey.pdf, 2015). It was developed to facilitate data sharing between different labs at AIBS. It applies a defined schema that was designed with the goal to facilitate software support while at the same time it aims at being intuitively understandable for the user (NWBh1_09_Keith_Godfrey.pdf, 2015). It utilizes object oriented design to provide extensibility and backwards compatibility (NWBh1_09_Keith_Godfrey.pdf, 2015).

The general objects used to store data are called data 'sequences', which represent a superset of several INCF types (NWBh1_09_Keith_Godfrey.pdf, 2015). The root sequence in the object-oriented hierarchy contains time representation and meta-data. Sub-classes represent increasingly refined data-types (NWBh1_09_Keith_Godfrey.pdf, 2015).

'Modules' are designed to represent the results of common data processing steps, e.g. spike sorting or image segmentation (NWBh1_09_Keith_Godfrey.pdf, 2015). A module

corresponds to a specific analysis process and expresses a defined interface to its data. Modules are also arranged in an object-oriented hierarchy.

The top-level of the file is explained in Figure 13 in Appendix I (adapted from NWBh1_09_Keith_Godfrey.pdf, 2015). All of the following formats build upon this basic layout and were adapted to the specific needs of the community.

'Borg' Format: Prototype of a common file format. The Orca format addressed many of the basic needs for a common file format that were defined by the community. However, it had to be extended and modified to accommodate the use cases and the general requirements defined in the 'what' document. A detailed description of Borg is attached in Appendix V. Borg takes up and integrates a variety of ideas from other file formats like the 'Kwik' format or the Svoboda lab file format (see Appendix V). It was the first version of a file format that was used to convert the example data sets.

The basic file organization at the top-level is equivalent to Orca (see above). The Borg format is based on 'TimeSeries' and 'Modules' which adopt the basic ideas of 'Sequences' and 'Modules' in Orca. However, Orca wasn't refined enough to satisfy the use cases and the specific requirements defined by the community in the 'what' document. The 'TimeSeries' and 'Modules' were extended and modified continuously during the process of converting the different example datasets to serve this purpose.

TimeSeries are organized in a strictly hierarchical manner. The root class is defined in a minimalistic way and is sub-classed to account for different modalities and data storage requirements (see Appendix V). Each TimeSeries has its own HDF5 group (folder), which contains all data and elements of the given TimeSeries. The most important components are 'data' and 'timestamps' (see Appendix V). Data represents the data

values and timestamps stores time information which is synchronized to the master clock of the experiment and stored in units of seconds to provide consistency throughout the experiment (see Appendix V).

Modules are used to store data, results and information about specific data processing steps (see Appendix V). Modules have their own folder which resides in /processing. A module contains one or more specific module interfaces, which are customized for the data processing step they represent (see Appendix V).

The top-level groups for Borg are the same as for Orca, just more precisely defined (adapted from Appendix V):

/general: Experimental metadata. Borg provides an interim solution for the metadata problem: While some common metadata fields are specifically defined and machine readable, the general approach is to use free-form text fields to store metadata.

/acquisition: Data streams recorded from the system. Bulky raw data can be stored in separate HDF5 files. In this case Borg contains the links to these data under acquisition.

/stimulus: Data pushed into the system (e.g. video stimulus, sound, etc.). To allow re-using stimuli in different experiments, template stimuli can be stored and used multiple times. Templates can also be HDF5-linked to a remote library file.

/epochs: Logically distinct segments of an experiment. Epochs contain specific acquisition and stimulus data. Different epochs can overlap. Time intervals, which should be ignored for various reasons can be marked within an epoch.

/processing: Intermediate analysis of data is organized in form of modules, which reside in this folder.

/analysis: Lab-specific and custom analysis. There are no restrictions on the form or schema, but the analysis data should be documented to make them shareable.

The file structure is illustrated in Figure 14 in Appendix I.

Borg addresses a variety of the specific concerns/requirements that the NWB workgroup identified:

- The metadata problem is solved in a preliminary way by the combination of a small number of explicit machine-readable fields and entries in free-text format. This solution was straightforward to implement and provides a suitable interim solution
- All time series are synchronized to a common experiment masterclock.
- The format is extensible without breaking backward compatibility. It allows users to add new datasets to existing TimeSeries or Modules or by defining new TimeSeries and/or Interfaces, which have to be documented to facilitate file sharing.

'NWB' Format: The final Product. The data files have been converted by the 'Borg' prototype in March 2015 and reviewed by data providers and tool developers. While there were only minor changes required to the format itself, concerns were raised about the methods used to specify and implement the format (Teeters, Godfrey *et al.,* 2015). The documentation was not sufficient to describe the implementation and some ambiguities left room for interpretation (Teeters, Godfrey *et al.,* 2015). Furthermore there

was the problem of managing extensions of the format, as they would require code changes in the implementation of the API (Teeters, Godfrey *et al.,* 2015).

To address these issues an API was developed that was based on a specification language that would describe the entities of the format in a JSON-like syntax, which is both machine and human readable (Teeters, Godfrey *et al.,* 2015). Other APIs, like the one for the NeXus scientific format (NeXus Scientific Data Format, 2016) or swagger (Swagger, 2016), inspired this approach, which provides a multitude of advantages.

The specification file is the only source that defines the format specification, independent from the rest of the API (Teeters, Godfrey *et al.,* 2015). It facilitates easy validation of the NWB files, already during the process of their creation through the API (Teeters, Godfrey *et al.,* 2015). Changes or extensions of the format only trigger changes in the specification file, which is practically independent of the rest of the API (Teeters, Godfrey *et al.,* 2015). This also makes it easy to implement the API in multiple programming languages. Thus it is straightforward for individual labs to adjust the format and the API to their individual needs. Extensions can be easily shared and can trigger alterations to the standard to keep the NWB format up-to-date with the needs of the community (Teeters, Godfrey *et al.,* 2015).

The NWB format specification file contains two sections: one describes the elements of the data model and the other one specifies the location where these elements are stored in the HDF5 file (Teeters, Godfrey *et al.,* 2015). An excerpt of the NWB specification file is attached in Appendix VI.

Currently, the write API is implemented in both, Python and Matlab. Documentation and information about the file format, the APIs and on-going efforts can be found at

Neurodata Without Borders | The Kavli Foundation, 2016 and Neurodata Without Borders, 2016.

Methods

To serve its purpose, the file format has to provide enough experiment specific information to satisfy all the different use cases on the one hand, while on the other hand it has to achieve a certain degree of standardization. To satisfy the use cases all the relevant information contained in the original data sets has to be incorporated into the new file format. To achieve standardization the file format should be on the other hand as compact as possible. In the extreme case a simple accumulation of all the elements of the five original datasets would certainly satisfy all the use cases but would not be generic at all. And it is certainly not desirable for future uses if the incorporation of new types of datasets leads to an explosion of new content in the file format. The studies suggested build on the previous findings for the individual datasets and address the following questions:

1. How diverse are the model data sets?
2. How are the original data represented in the new file format?
3. How well does the new data format generalize?
4. How does the new data format compare to existing ones?
5. To what extent does the new data format fulfill the requirements defined by the workgroup?

Investigation of the Diversity of the Model Data files

To address question 1., a thorough study of the datasets that highlights the parts that are similar and the parts that are clearly different was conducted. The NWB workgroup has identified a variety of different sub-categories that the researchers want to represent in the new file format: Experiment and animal information, intracellular electrophysiology, extracellular electrophysiology, optophysiology, sensory stimuli, simple optogenetic stimuli, behavioral events and pharmacology during the experiment. Mapping the different parts of the original datasets onto these categories will provide some rough guidelines for categorizing the datasets. Ideally all categories should be represented by the five example datasets to guarantee that everything is covered. If there are overlaps between the five different dataset these will be further analyzed to classify the extent of similarity.

The processing stage of the individual parts of the data sets (from raw data to highly modified data) and the individual analysis goals will provide further criteria for comparison of the different datasets.

The results of this investigation provide some important background information needed for other studies, especially question 3.

Mapping of the Original Data Sets to the New File Format

In the next step, the mapping of the components of the original files to the new data format will be investigated. This addresses questions 2 and 3.

The datasets are converted to the new data format by mapping their contents to individual fields in the new datasets. The original datasets should be represented completely in the new data format. The new data format contains pre-defined fields,

which aim at standardization. Apart from that it allows to add custom entries that reflect individual requirements for a given dataset. The mapping of the contents of the original datasets to their representation in the new data format will reveal the degree of required custom fields specifically added for the individual datasets on the one hand and the number of unpopulated pre-defined fields in the new data format. These quantitative measures directly relate to the degree of standardization: A large proportion of customs fields for the datasets could indicate a lack of pre-defined fields and an insufficient degree of standardization whereas a big amount of empty fields might point to unnecessary content in the new file format and fields which are too specialized to be generic.

However, these numbers have to be carefully evaluated on the basis of the findings for question 1. These results will strongly depend on the degree of diversity of the model data sets and have to be evaluated based on this parameter.

Comparison with Existing File Formats

A thorough assessment of the subset of existing file formats described above was conducted. This determined which requirements for a common format are addressed by the different file formats. Some important points to investigate are: ways to represent diverse data, degree of standardization, extensibility and ease of use. The results were compared to the new file format and were used to benchmark its performance.

Metric Evaluation of the File Format

The workgroup identified a variety of possible evaluation criteria. A subset of these is listed in Table 2 in Appendix II, which was used to develop a metric to assess the

progress of the development process. Some of these criteria define minimum requirements to make the data format useful:

- Ability to store/represent ephys and ophys data
- Open source
- Extensibility
- Python support
- Matlab support
- View data without coding

Others are desirable near-term goals:

- Robust tool ecosystem
- Ease of importing new datasets
- Additional applications for viewing
- Java support
- C/C++ support

Others are performance measures:

- High read bandwidth for specific data
- High read bandwidth for session
- Storage

The metric captured these three categories in three numbers to yield a measure of the format x/y/z where x represents the essential requirements, y the near term goals and z the performance. The different criteria will be translated into numbers as suggested in Table 3 in Appendix II.

Chapter III

Results

This chapter presents the results we obtained by using the different methods described in chapter II.

Diversity of the Data Sets

This section investigates the diversity of the model data sets. First we map the datasets onto the different categories of the ‘what’ document, which is provided in Appendix VII. Then we take a look at the content and complexity of the data sets to further differentiate them with respect to each other.

Mapping the Data Sets onto the ‘What’ Document

The ‘what’ document is a collection of data elements, which should be represented in the new file format to match the needs of the community. Leading researchers in the field created it during and after the first Hackathon. It is structured into 8 distinct modules:

- Module 0: Experiment and Animal Information
- Module 1: Intracellular Electrophysiology
- Module 2: Extracellular Electrophysiology
- Module 3: Optophysiology
- Module 4: Sensory Stimuli

- Module 5: Simple Optogenetic Stimuli
- Module 6: Behavioral Events
- Module 7: Pharmacology

The complete 'what' document is attached in Appendix VII.

Table 4 in Appendix II shows how the different example datasets map onto the modules of the 'what' document.

Two categories, intracellular electrophysiology and pharmacology are not linked to any of the four datasets. Intracellular electrophysiology is addressed by CRCNS pvc-6, which is not included in this research project. Pharmacology is the only module, which is not covered in any way at this point in time. Two modules are only addressed by one dataset: Simple Optogenetic Stimuli are only included in CRCNS alm-1 and Optophysiology is so far only represented by CRCNS ssc-1. Experiment and Animal Information, Extracellular Electrophysiology, Sensory Stimuli and Behavioral Events are mapped to multiple datasets.

Processing Steps and Content of the Data sets

Next we look at the content of the datasets and address the following questions:

- How 'complete' are the data? Are all stages of analysis contained and documented in the file (provenance)?
- How much information is encoded in the data files? How many entities do they have? How complex is the file?

CRCNS ret-1 is the most 'lightweight' data set. It does not provide raw data in any form. The only data it contains are processed spike data without any documentation of their

origin. Apart from the processed data, ret-1 has only eight fields for metadata and ten fields to describe the stimuli. The maximum depth of the file hierarchy is three levels. Thus CRCNS ret-1 is at the low end for both completeness and complexity.

CRCNS hc-3 is at the high end in terms of completeness: It provides 10 different types of data which represent up to five processing steps, though the main data file does not contain any direct link to the raw data. Extensive metadata are provided in three different zip files. Thus the complexity is intermediate.

CRCNS alm-1 provides raw data and processed data for electrophysiology but only raw data for the behavioral events. Thus the completeness is intermediate. The main data file of alm-1 is fairly complex: it provides eight different datasets and has ~40 different fields. The data hierarchy is up to four levels deep.

CRCNS ssc-1 provides raw data and processed data for both the optophysiology and the behavioral events. Thus it is fairly complete, though the data analysis doesn't go quite as far as for hc-3. It has the most complex data file among the four datasets: The main data file contains fifteen different types of data, ~ 50 different fields and up to seven levels in the data hierarchy.

Figure 15 in Appendix I summarizes the findings for the four different datasets.

Mapping of the Data Sets to the new File Format

This section investigates how the individual datasets are mapped onto the NWB file format. The mapping plots will show how the individual parts of the main data files are translated into the NWB file. As some of the files are fairly complex it is not

straightforward to create a detailed mapping plot that is easy to read and understand. As a consequence we simplified some parts of the plots, which are explained below:

- In general the main focus of the plots is to illustrate how the main datasets and the key parameters are mapped onto the NWB file. The mapping of generic fields 'file_create_date' or simple descriptions have been left out
- The /general folder contains the metadata in the NWB format. We list parts of it to illustrate some of its contents but don't map any entries to their origins. The current solution for the metadata is based on free text format and preliminary in nature. Mapping all of the metadata would be complicated on the one hand and not very insightful on the other as NWB currently does not really aim at solving the metadata problem. It just provides an interim option to include them in the file.
- The /epoch folder provides the possibility to define subsections of the data and assemble different data which belong to the same logical entity. The datasets are typically only links to other parts of the NWB file. We don't explain the origin of the datasets linked in the /epoch folder as this has already been done elsewhere. We mainly provide the mapping of the 'tags' field, which explains the meaning of the epoch subunits.
- /acquisition, /processing and /stimulus hold the major parts of the data. The maps have a strong focus on these parts
- Alm-1 and ssc-1 make heavy use of key-value pairs. That means in this case that the key fields are human readable, contrasting to the real data folders that are typically labeled with numbers. To make the maps easier to understand we

connected the keys (and not the data folders) to the matching NWB data entries and pointed the data folders to their keys to make the link clear. In fact this means that both, the data and their labels are mapped onto the part of the NWB file that is connected to the keys in the map.

CRCNS hc-3

Figure 16 in Appendix I shows how the hc-3 dataset is mapped onto the NWB file format.

Most of the data in the main data file are included in the NWB file except for ec*.mm and ec*.m1m2. These are highly specialized analysis files, which have been derived in close interaction with the KlustaSuite software. Currently Klusta is not directly supported by NWB, so they have been left out in the NWB file. If this should change, these data files could be included in the /analysis folder.

The resulting NWB file seems unlike more complex than the original main data file which is in essence a collection of plain data files in a session folder. This comes with the benefit that the resulting new file is a lot easier to understand for the general user as the arrangement of the data files within the NWB file already makes a statement about their meaning and they are furthermore put in place together with matching parameters and descriptive elements.

Hc-3 does leave a couple of fields blank in the NWB file, but does only need one custom field.

The following parts of the NWB file are empty for hc-3:

- /acquisition/images and /acquisition/timeseries: this indicates that there are no raw

data present in the original file.

- /analysis: no specialized analysis files are included at this moment in time
- /general/devices: no information about used devices in the file
- /stimulus/presentation and /stimulus/templates: hc-3 does not contain any stimuli

The following is a custom field:

- /processing/shank_0/EventWaveform/waveform_timeseries/sample_length: length of the individual waveforms

CRCNS alm-1

Figure 17 in Appendix I shows how alm-1 is mapped onto the NWB file format. The mapping is fairly complete. All data files and the majority of the descriptive elements are included in the NWB file. The result of the mapping is a file of comparable complexity, which is more intuitive and in major parts easier to understand than the original one. Alm-1 leaves little empty space in the NWB file and does require a couple of custom fields.

The following parts of the NWB file are empty for alm-1:

- /acquisition/images: indicates that there are no raw images included in the main file
- /stimulus/templates: the experiment does not utilize templates for the stimuli.
- /epochs/Trial_n/description: epochs have no descriptions
- /epochs/Trial_n/ignore_intervals: no ignore_intervals in any epoch

The following are custom fields:

- /epoch/Trial_n/units: links electrophysiological units to a given trial

- /processing/Units/EventWaveform/unit_n/sample_length: length of the individual waveforms
- /processing/Units/UnitTimes/CellTypes: provides cell type information for the electrophysiological units
- /processing/Units/UnitTimes/ElectrodeDepths: gives the electrode depths for the electrophysiological units
- /processing/Units/UnitTimes/unit_n/trial_ids: indexing information. Links a specific electrophysiological unit with its trials.

CRCNS ssc-1

Figure 18 in Appendix I shows how the ssc-1 dataset is mapped onto the NWB file format.

Just like for alm-1 the mapping is in essence complete and results in an NWB file with comparable complexity. In terms of readability the NWB version has some advantages compared to the original file as also in this case the NWB file hierarchy helps to make the file more intuitive and easier to understand. The mapping of ssc-1 leaves almost no empty space in the resulting NWB file. Ssc-1 needs a couple of custom fields to realize a sufficiently complete mapping onto the new file format.

The following parts of the NWB file are empty for ssc-1:

- /analysis: no specialized analysis files are included at this moment in time
- stimulus/templates: the experiment does not utilize templates for the stimuli.
- /epochs/Trial_n/ignore_intervals: no ignore_intervals in any epoch

The following are custom fields:

- /epochs/Trial_n/ROI_planes: lists ROI_planes for the given epoch
- /epochs/Trial_n/ROIs: lists ROIs for the given epoch
- /processing/ROIs/DfoverF/fov_n/trial_ids: indexing information. Links fluorescence data with the matching trials.
- /processing/Whisker/BehavioralEpochs/pole_touch_*/ /kappa_max_abs_over_touch: additional whisker curvature information for touches

CRCNS ret-1

Figure 19 in Appendix I shows how ret-1 is mapped onto the NWB file format. The mapping increases the file complexity significantly. However the NWB file seems user-friendlier as it provides more structure and descriptive elements.

Ret-1 leaves a couple of fields blank in the NWB file and needs a remarkable number of custom fields despite the small amount of information and complexity in the original file. The following parts of the NWB file are empty for ret-1:

- /acquisition/images and /acquisition/timeseries: this indcates that there are no raw data present in the original file.
- /analysis: no specialized analysis files are included at this moment in time
- /general/devices: no information about used devices in the file
- /stimulus/templates: the experiment does not utilize templates for the stimuli.
- /epochs/Trial_n/description: epochs have no descriptions
- /epochs/Trial_n/ignore_intervals: no ignore_intervals in any epoch

The following are custom fields:

- /processing/Cells/UnitTimes/cell_n/stim_n: custom datasets need to be defined to accommodate the fact that multiple spike time files belong to an individual cell

The following are additional custom fields to describe the reconstructed stimuli in the /stimulus section:

- /stimulus/presentation/meister_dx
- /stimulus/presentation/meister_dy
- /stimulus/presentation/meister_x
- /stimulus/presentation/meister_y
- /stimulus/presentation/pixel_size

Comparison with Existing File Formats

The following section provides a short overview of how the NWB format compares to the existinging formats for neurophysiological data storage that served as direct inspiration for the NWB format. The focus is on outlining similarities and differences and to make some brief suggestions about what could still be adapted from the other formats to improve the NWB format. Table 5 in Appendix II lists to what degree key features are realized in the various file formats. Details for the individual file formats are explained below.

Kwik Format

The Kwik format is one of the formats that provided direct input to the NWB format. Specifically, NWB adapted the modular approach of Kwik. This provides user-friendliness and it is also a core feature that gives NWB flexibility and extensibility.

NWB has a fairly well defined structure with a separate section for the metadata, raw data, stimuli and processed data. This gives the user a rough idea about where to find individual parts of a data file. Kwik is more limited in this aspect. Furthermore NWB provides provenance information and the possibility to define subsections of the data, which Kwik is lacking. Kwik separates bulky raw data from the main data file to keep this file compact. NWB is not going that far. It encourages providing external links for the raw data but this practice is not enforced. Kwik is to date a specialized format for extracellular recordings whereas NWB has been developed to support a wide range of data. The biggest advantage however that Kwik provides is that it works together with KlustaSuite, a widely used application for spike sorting. This is one essential milestone that NWB still has to achieve to establish wide range acceptance among the extracellular electrophysiology user group.

Svoboda Lab Data Format

The Svoboda Lab format also provided some inspiration for the creation of NWB. NWB adapts the time representation of the Svoboda Lab data format. Also, its TimeSeries data representation aims at creating an equivalent to the Svoboda key-value pair approach. Compared to the NWB format the Svoboda Lab format is incredibly compact and optimized to represent data files which a high volume of content. This goes at the expense of user-friendliness, as the Svoboda Lab format is not intuitively understandable and consequently rather difficult to use for naïve users. Also, the Svoboda Lab format does not provide provenance information and does not directly define subsections of datasets in a flexible way. Instead it currently uses indices to access specific subsections called trials. Indices can greatly enhance access to subsections of big

datasets. This is something that NWB should consider to improve performance and handling of voluminous datasets.

Nix Format

Nix is another format that provided key ideas for the creation of NWB. Nix tags inspired the creation of epochs for NWB. Like Nix, NWB provides unit information for the datasets to support easy plotting of the data. The Nix block and group structure is very similar to the top-level folders with subfolders in NWB. Both Nix and NWB have elements to provide provenance information though NWB goes a little bit further in this aspect. Different to NWB Nix is based on C++. This creates some hurdles for the user community as C++ is more challenging for users who want to get involved in the development and there is the potential for build issues, which can be a real challenge for the average user. NWB is currently implemented in Python and Matlab, which is much easier to handle for a variety of reasons. Nix is more generic as it allows importing of practically any data and to this point it is more developed than NWB. It has a more specific approach to include metadata, it provides wide language support and it comes with tools for benchmarking and validations as well as with extensive documentation and tutorials. Compared to Nix, NWB still has to catch up in terms of language support, documentation and tools provided.

LBNL Brain Format

There are also a variety of similarities between NWB and the LBNL Brain Format. Both utilize a JSON specification, which makes it easier to comprehend and extend the file format. They both include units for the data for easy plotting and support

modular design. They also both provide the possibility for data annotation, though the epoch approach of NWB is more flexible and probably easier to comprehend. The Brain format is also very specifically developed for ECoG data, different to the NWB format, which supports a variety of different use cases. Different to NWB, the Brain format does not provide any provenance information. However, the Brain format includes mechanism for file validation already upon creation. NWB could definitely benefit from more development in along these lines.

Metric Evaluation

The following is a metric evaluation of both Borg (the interim format) and NWB (the final outcome) by using the criteria defined in the methods section of the last chapter. The results give an exemplary snapshot of the performance of the two file format stages and they illustrate the improvements of NWB compared to Borg. The remaining shortcomings for NWB serve as an additional source for future improvements.

Borg format

Table 6 in Appendix II provides a scorecard for the Borg format. The individual entries are explained in the following:

1. The format was designed to do exactly that and the translation of the example data sets demonstrates that it works.
2. The write API for the Borg format is open source.
3. The data format itself can easily be extended, mainly through adding custom

fields for small additions or new new sub-classes of timeseries and new modules for completely new entities. Adding custom fields is very easy – they simply add additional lines in the conversion scripts. However, these new timeseries and module extensions are not straightforward, as they involve API changes.

4. HDFView provides this functionality.
5. Python write API, no read API.
6. No Matlab APIs.
7. No third party tools are directly supported at this point in time.
8. Importing new datasets with Borg is not exactly an easy task. The time it takes to import a new dataset is of course strongly dependent on the size and complexity of the dataset. However, 5-10 days is a fair estimate for a developer to accomplish this task and it likely takes quite a bit longer for an average user.
9. Borg files can be viewed in HDFView. Additionaly a GUI application for Matlab import was provided.
10. Java is not supported.
11. C/C++ is not supported.
12. To test the performance for loading single datasets, four individual datasets have been loaded into Matlab. Each of them was extracted from the original data file and from the Borg data file. The times needed for loading the datasets into Matlab have been tracked. This is repeated ten times for both files and the average times for opening the datasets are compared. The exemplary datasets which have been picked are:
    - A spike waveform dataset from CRCNS hc-3. This is a large dataset in

binary format in the original file.

- A spike waveform dataset from CRCNS alm-1. This is a large dataset included in a Matlab struct.
- Behavioral Data from CRCNS ssc-1. This is a small dataset included in a Matlab struct.
- A reconstructed stimulus from CRCNS ret-1. This dataset is reproduced from a binary file with random numbers for the original data file. It is directly included in the Borg file.

Table 7 in Appendix II lists the loading times and the score for the four datasets. The total score in this category is 4, which is the average of the four individual scores.

13. To assess the performance of Borg for a typical session, we took example data analysis scripts provided for the four datasets and reduced their content to the operations which involved loading and querying of datasets. This scripts are:

    - 'Demo_get_performance.m' (script_1): a script, which belongs to CRCNS alm-1. This is a rather short script which only queries smaller stimulus and behavioral datasets.
    - 'Demo_get_trial_aligned_raster_PSTH.m' (script_2): another script for CRCNS alm-1. This script queries data all over the file structure, including spike times.
    - 'plotTrial.m' (script_3): a script for CRCNS ssc-1. Queries many different datasets of different sizes all across the file.
    - 'analysis_example' (script_4): for CRCNS ret-1. This script queries both

small and big datasets for ret-1 including a reconstructed stimulus. This dataset is reconstructed from a binary file for the original data compared to being simply extracted from the file in the case of the Borg file.

We ran this scripts on both, the original data files and the Borg file. The running times for both the original data file and the Borg file were tracked. This was repeated ten times and the average times for all of the runs have been compared for the two files. The results together with the scores for all scripts are listed in Table 8 in Appendix II.

The total score is 2.75, which is the average of the four individual scores.

14. Table 9 in Appendix II compares the size of the original files with their Borg counterpart. The total score is an average of the four individual scores.

    The total score is 2.5, which is the average of the four individual scores.

The total score for Borg is 29.25 out of 67. The total score for the individual areas minimum requirements (items 1-6 of the scorecard)/near-term goals (items 7-11)/ performance (items 12-14) is 17/3/9.25 out of 21/16/30.

NWB format

Table 10 in Appendix II provides a scorecard for the NWB format. The individual entries are explained in the following:

1. The format was designed to do exactly that and the translation of the example data sets demonstrates that it works.
2. The write API for the NWB format is open source.
3. The specification language provides an easy means for extensions without

triggering API changes

4. HDFView provides this functionality.
5. Python write API, no read API.
6. Matlab write API, no read API.
7. No third party tools are directly supported at this point in time.
8. Importing new datasets with NWB is easier than with Borg. The new API is simpler, the scripts in general need fewer API calls and the specification language makes the format easier to understand. These simplifications are expected to roughly half the time required to perform this task.
9. NWB files can be viewed in HDFView. Additionaly a GUI application for Matlab import was provided.
10. Java is not supported.
11. C/C++ is not supported.
12. The approach is the same as for Borg. The results are listed in Table 11 in Appendix II.

    The total score in this category is 3.5, which is the average of the four individual scores.
13. The approach is the same as for Borg. The results are listed in Table 12 in Appendix II.

    The total score is 4.25, which is the average of the four individual scores.
14. Table 13 in Appendix II compares the size of the original files with their Borg counterpart. The total score is an average of the four individual scores.

The total score is 6, which is the average of the four individual scores.

The total score for NWB is 37.75 out of 67. The total score for the individual areas minimum requirements (items 1-6 of the scorecard)/near-term goals (items 7-11)/ performance (items 12-14) is 19/5/13.75 out of 21/16/30.

Chapter IV

Discussion

This chapter discusses the results we collected in chapter III as well as the limitations of our study. It finally comes to a conclusion about the research hypothesis.

Analysis of the Results

This section provides an analysis and a discussion of the results described in chapter III.

Diversity of the Datasets

Overall, the example datasets model the majority of the ‘what’ document. The only modules not addressed are ‘Intracellular Electrophysiology’ and ‘Pharmacology’. Two modules are only mapped by one dataset: ‘Optophysiology’ and ‘Simple Optogenetic Stimuli’. This could indicate a lack of diversity in these fields as the representation of the modules might just be ‘handcrafted’ for these respective datasets without really representing the diversity in the field.

The other modules are represented by multiple datasets and discussed in detail below.

- Experiment and animal information:

  This module contains the metadata for each dataset. This module is ‘naturally diverse’ as most experiments need some very specific fields for their description. The NWB format currently only provides a preliminary solution for the metadata

problem, which allows representing diverse metadata but does not yet address the actual challenge. Thus assessing the performance of the NWB with respect to the metadata it is not really insightful at this point in time and we will accordingly skip any more detailed diversity analysis for this module.

- Extracellular electrophysiology:

  CRCNS ret-1, CRCNS alm-1 and CRCNS hc-3 contain extracellular electrophysiology data. There are a lot of similarities in terms of data types included in the files, which is expected for this module. However, there is also some diversity in terms of complexity and completeness of the different datasets, which is illustrated in Fig 16. These differences are in particular realized in the Extracellular Electrophysiology module. All of the three datasets contain a voltage trace though with very different levels of detail: ret-1 only provides spike times, alm-1 has spike times, waveforms and raw data whereas hc-3 contains spike times, waveforms, LFP data, PCA features used to do spike sorting and classification data for each spike along with some additional files used for or generated by the analysis process. The difference in terms of complexity is also quite significant with ret-1 and alm-1 being on opposite ends of this parameter. These differences account for some diversity for this module.

- Sensory Stimuli:

  CRCNS ret-1, CRCNS alm-1 and CRCNS ssc-1 contain sensory stimuli. There are a total of four different stimuli: an auditory cue, a pole in reach, a water reward and reconstructed visual stimuli. Table 14 in Appendix II shows which stimuli belong to which dataset. Alm-1 and ssc-1 have some overlap as they both

provide a pole in reach and an auditory cue as stimuli. Ssc-1 has a water reward in addition. The visual stimuli are unique to ret-1. The number of different stimuli is small and three of them, auditory, pole and water, are simple stimuli represented as events or intervals. The visual stimuli of ret-1 however are distinctively different as they are reconstructed images, which have been generated by the use of a binary file with random numbers. Thus, there is some modest diversity for the the stimuli module.

- Behavioral Events:

  Four different behavior data (Head position, licks, pole touches and whisker position) are contained in three different datasets, CRCNS hc-3, CRCNS alm-1 and CRCNS ssc-1 as is shown in Table 15 in Appendix II. While this is a small number of behaviors they are at the same time remarkably different. The head position, which is recorded in hc-3 represents 14 different behavioral tasks. The data originate from two LEDs attached to the subjects' head. The licking behavior, which is observed in both alm-1 and ssc-1 reports the decision of an animal after being presented a pole in a certain position. It is encoded in the output of a photodiode. The pole touches and whisker position, which are part of ssc-1 are derived from the analysis of whisker videos. In summary, there is diversity in the behavioral events module represented by a small number of fairly different behaviors.

Despite the small number of datasets a wide range of the requirements is covered though there are also some weaknesses like e.g. for the Optophysiology module which is only represented by one dataset or the Pharmacology module which is not represented at all as

well as some similarities of the datasets.

Mapping of the Datasets to the new File Format

Table 16 in Appendix II gives an overview about the empty folders and fields found in all four NWB files. The different entries are discussed below.

- /acquisition/images, /acquisition/timeseries, stimulus/presentation: It seems natural that not all datasets have raw images and raw timeseries data or stimulus data as this depends on the experiment design. In addition data files should provide some link to rawdata, a description of the epochs (if epochs are used) and some device information. All of these folders seem to be an important part of the data format despite being left blank by some of the example datasets.
- /analysis, /epochs/epoch_n/ignore_intervals and /stimulus/templates could be unnecessary blanks as they have few or no entries with the current datasets and are not necessarily an essential part of an experiment. The analysis folder has only been used by alm-1 so far. This is an add-on folder, which labs can use to include very specific data and information for their data analysis. It's possible that it will turn out to be redundant if few or no labs make use of it, but it's not straightforward to come to a conclusion about this with the small number of example data sets. /epochs/epoch_n/ignore_intervals and /stimulus/templates have not been used at all with the given datasets. Thus they are more likely to be fields, which are too specialized for generic use and likely left blank for a large proportion of translated datasets.

Table 17 in Appendix II gives an overview about all custom fields found in the NWB

files. The custom fields are indexed with a number code to simplify the discussion.

- Fields 1 and 2 are practically identical as they both appear in conjunction with the spike waveforms. Alm-1 and hc-3 are the only two datasets, which include spike waveforms. The fact that they both add this field likely indicates that this should be in general part of the file format.
- Fields 3-7 provide indexing information for alm-1 and ssc-1. They link ROIs and units to their trials and create a means for easy extraction and analysis of the data. While this seems to be specialized information for the Svoboda lab data it could however be a useful extension to the file format to provide some space in the NWB file for indexing. It is expected that future datasets get even bulkier than the given ones and indexing could be a vital strategy to simplify the navigation of the resulting data files.
- Fields 8-10 seem to be more specialized fields at this moment in time. Thus it might be justified to add them as custom fields.
- Field 10 seems unique for among the given example datasets but it might in general be a good extension to allow multiple datasets for a given unit.
- Fields 11-16 are all required to adequately describe the data associated with the reconstructed stimuli of ret-1. The file format does clearly not provide an ideal structure for this use case. To address this, it seems reasonable to include a separate timeseries for this type of stimulus in the specification language and get rid of all these custom fields.

Overall the file format seems to be a good match for the four example datasets. It seems to provide a rather ‘slim’ approach as there are very few blanks opposing a number of

custom fields. The biggest mismatch occurs for the mapping of CRCNS ret-1, as this has the largest number of blank fields while at the same time it requires the largest proportion of custom fields.

Based on the four example datasets it also seems that the new file format achieves a reasonable degree of standardization. The majority of the file content can easily be mapped onto the new format without producing too many unnecessary or specialized fields for the individual data file. Extensions for indexing and reconstructed stimuli seem a straightforward improvement to further improve the mapping and to provide a better means to handle bulky datasets.

However, the small number of example datasets and the limitations in diversity impair the results. This is especially true for the field of intracellular electrophysiology, which has not been addressed at all in this study and for the field of optophysiology, which is only represented by one dataset. Consequentially, it seems pre-mature to come to a general conclusion for the question of standardization.

Comparison with Existing File Formats

Table 5 in Appendix II lists to what degree key features are realized in the various file formats. Three of the data formats; Kwik, the Svoboda Lab Data Format and the LBNL Brain format have been developed for a fairly specific community. It is not surprising that they are less generic than NWB. Kwik has the unique benefit of supporting KlustaSuite. This is the most obvious issue for NWB, which should be resolved in the nearer future. Apart from that NWB addresses most of the weaknesses found for the other file formats, though there is certainly room for improvement.

The epoch approach of NWB, which is used to define sub-sections of the data, has the advantage of being very user-friendly and human readable with the downside that many subfolders and additional links are created. It's possible that this might prove burdensome for some datasets in the future. Nix and LBNL Brain both provide an alternative approach, which could be consulted to work out a different solution. Indexing as practiced in the Svoboda Lab Data Format might be another promising solution to retrieve excerpts of large datasets.

The metadata solution for NWB is preliminary. Again, the Brain and the Nix format could be used as a resource as the metadata solution is more developed for these two formats.

File validation is another important part that is currently underdeveloped in NWB. It is highly desirable to implement a procedure, which already validates the file content during its creation as demonstrated in the LBNL Brain format. Furthermore it is also important to have tools in place to validate existing files, as it is e.g. the case for the Nix format.

In general, NWB is still fairly young and in some aspects rather sparsely developed. Further parts of NWB, which need more development, are support of various languages like e.g. Java or C/C++, more documentation, tests, tutorials and additional tools

Metric Evaluation

The metric is divided into three subsections: Minimum requirements, near-term goals and performance, which are discussed in detail below.

- Minimum Requirements: NWB shows some progress compared to Borg and fulfills almost all of the minimum requirements. Borg was lacking the read API for Python, both APIs for Matlab and it wasn't quite straightforward to make

extensions, as they would require changes of the API itself. NWB introduced a JSON specification language for the API, which makes extending the format unlike easier and it added a native Matlab write API. NWB is still missing both read APIs for Python and Matlab. While this is certainly important it's not a critical tool for usage of the NWB files. Both Python and Matlab provide hdf5 libraries, which can be used for importing and handling NWB files. Thus, it seems reasonable to move the read APIs to the category 'near-term goals' and consider NWB as complete in terms of minimum requirements.

- Near-term Goals: Little has been achieved in terms of near-term goals for Borg and also NWB shows only modest progress in this area. Borg comes with one additional Matlab based tool that facilitates data import and viewing in Matlab and it earned a low score for ease of importing new datasets into Borg. While it is feasible, though time consuming for a developer to work with the Borg API, it seems challenging for an average user. NWB improved in this point as the JSON specification makes it significantly easier to understand and use the NWB API. But NWB still leaves a lot of room for improvements in this category: So far no third party tools are supported. This should be addressed in the nearer future as some of them like e.g. KlustaSuite are quite important for the community. Java and C/C++ support are also important items on the list to make NWB suitable for more potential users.
- Performance: Both Borg as well as NWB have fairly modest scores in terms of performance. Borg did very poorly in terms of storage requirements, as three out of the four Borg files are significantly bigger than the original files. It shows

fairly noticeable performance issues when it comes to reading single datasets into Matlab, which is even more pronounced on the session level. For NWB more care was taken about the appropriate data format and about avoiding duplicate datasets within the files by using links. This improved the file sizes quite significantly and also helped to some degree with the read performance issue. The final performance scores for NWB still seem fairly poor, but they have been derived in a very specialized way, which has to be taken into account. CRCNS ret-1 shows the most striking discrepancy in terms of file size. This is due to the fact that the reconstructed stimuli are provided in NWB as an externally linked stimuli library, which requires extra storage compared to the original data file, which comes with a binary file of random numbers and a method to reconstruct the stimuli whenever they are needed. This creates a large overhead if a single file is burdened with a large collection of stored stimuli datasets. However if many data files share a small number of stimuli datasets this would only result in a modest increase for the storage requirements. Also, storing the datasets instead of reconstructing them every single time as needed also results in performance improvement as can be seen for both the single dataset read-in as well as the session performance. All performance studies have been performed within Matlab. This is one-sided on the one hand but seems adequate on the other hand as all of the original datasets are typically viewed and processed in this environment. The overall conclusion is that the read functions for hdf5 within Matlab are significantly less efficient than the read functions for both native Matlab and binary data. There is little room for improvement in terms of changing the NWB files and most of it has probably

already been realized with the changes made as compared to Borg. However, a read API for Matlab has the potential to successfully address this issue. Matlab provides low-level hdf5 functions which are more performant but not straightforward to use for an average user. By carefully utilizing these functions and wrapping them up in a read API, the performance might improve without adding significant difficulties for the users.

Study Limitations

One limitation is certainly the fact that the final version of the NWB file format is not necessarily complete in all aspects. The NWB project schedule was incredibly tight and consequently the pace of development was accordingly fast. The final release of the NWB format should certainly be regarded as work in progress with respect to multiple aspects like e.g. the metadata or third party tool support. This hinders studying these aspects and also limits the significance of the results.

Also, the number of model datasets is rather small and consequently all the studies are small-scale studies. Despite the fact that these ‘high value’ datasets were chosen with great care with the goal to represent the plurality of existing data in the field, the four datasets which contribute to this study might not be sufficient to adequately predict how the file format will handle all types of present and future datasets.

Some of the evaluation criteria in the metric cannot be objectively measured and thus contain a human component. Consequently the absolute number might lack precision. This is true for the following criteria:

- "Extensibility": while it is objective to decide if the format can be extended, however, it is subjective to judge how easily this is achieved.
- "Ease of importing new data sets": These numbers are based on the experience of a small group of people and can certainly vary a lot among different user groups

The two read performance studies have been performed within Matlab. This is one-sided on the one hand but seems adequate on the other hand as all of the original datasets are typically viewed and processed in this environment. However it should be mentioned that choosing a different environment is likely to alter these results.

Last but not least it has to be mentioned that the actual reception of the file format is not necessarily predicted by this study. The user reception is probably the most important factor, as the users finally will decide if this file format will achieve its purpose. The success of this effort depends on a lot of unpredictable factors, like the political climate in the science community, the type of data coming up, etc. and only time will finalize the outcome of the NWB initiative.

Conclusions

The hypothesis of this research project is that this newly created file format is suited to become the common file standard within the community of cell-based neurophysiology. This hypothesis is supported by four major questions:

1. Is the new file format able to represent the diversity of existing data in this field and is flexible enough to accommodate the output of future experiments in this area?
2. Does the NWB format show a sufficient degree of standardization?

3. Does the new format overcome weaknesses of (selected) existing file formats?
4. Does the NWB format fulfill the requirements for the new common file format as defined by the NWB workgroup?

The NWB file format is working very well with the four model datasets used for this study. It is easily extensible and it also seems to reach a reasonable degree of standardization. However, the results for question 1 and 2 are impaired by the fact that the four model datasets are not quite diverse enough to allow for far-reaching conclusions. The NWB file format overcomes the major weaknesses of the other most promising file format candidates in the field except for the fact that it does not yet support KlustaSuite, an important application of the community of extracellular electrophysiology. The new file format also fulfills the minimum requirements reasonably well though it leaves some room for improvements with respect to performance and desirable near-term goals.

Altogether the results of this thesis show very well how much the NWB format has already achieved despite the short time of its development. However, due to the small sample of datasets and the resulting limitations in diversity we have to reject our hypothesis at this point in time. Future use and developments will show if the NWB format is able to continue the positive trends found in this research project and if it will fulfill its purpose to become a widely accepted uniform data standard within the research community of cell-based electrophysiology.

Appendix I

Figures

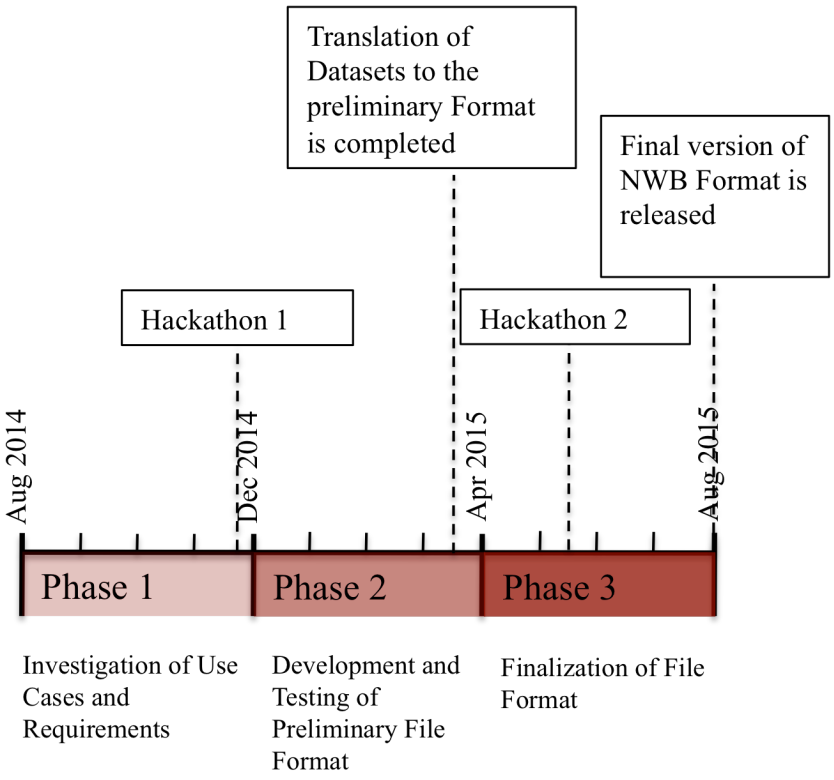


Figure 1. Timeline for the NWB project. Overview over the different phases and milestones.

ec*.dat	raw data or "wideband data"
ec*.xml	configuration file used with neuroscope software
ec*.mpg/ec*.m1v	video recording of animal position during the experiment
ec*.led	timing of LED synchronization light in mpg movie
ec*.whl	position of animals during the session, extracted from mpg/m1v file
ec*.eeg	LFP data, low pass filter output of .dat
ec*.fil	high pass filter output of .dat
ec*.threshold	threshold for spike detection in units of RMS noise level
ec*.res	time of each spike
ec*.spk	waveform for each spike
ec*.fet	features used for spike sorting generated from the spike waveforms
ec*.clu	cluster number for each spike after spike sorting
ec*.m1m2	auxiliary file created by the Klusters program
ec*.mm	auxiliary file created by the Klusters program

Figure 2. Comprehensive overview of all data files associated with the data. The data files for hc-3 represent different stages of data processing.

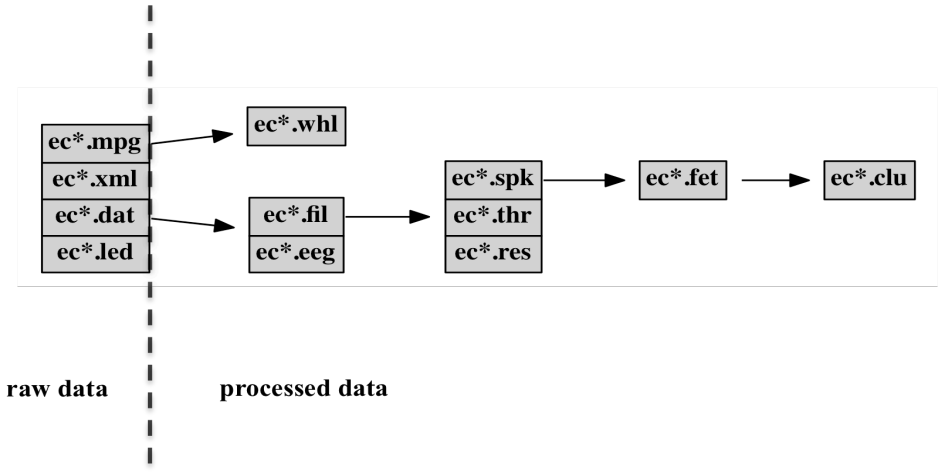


Figure 3. CRCNS hc-3 raw data and their products. The graph shows how the processed data are derived from the raw data.

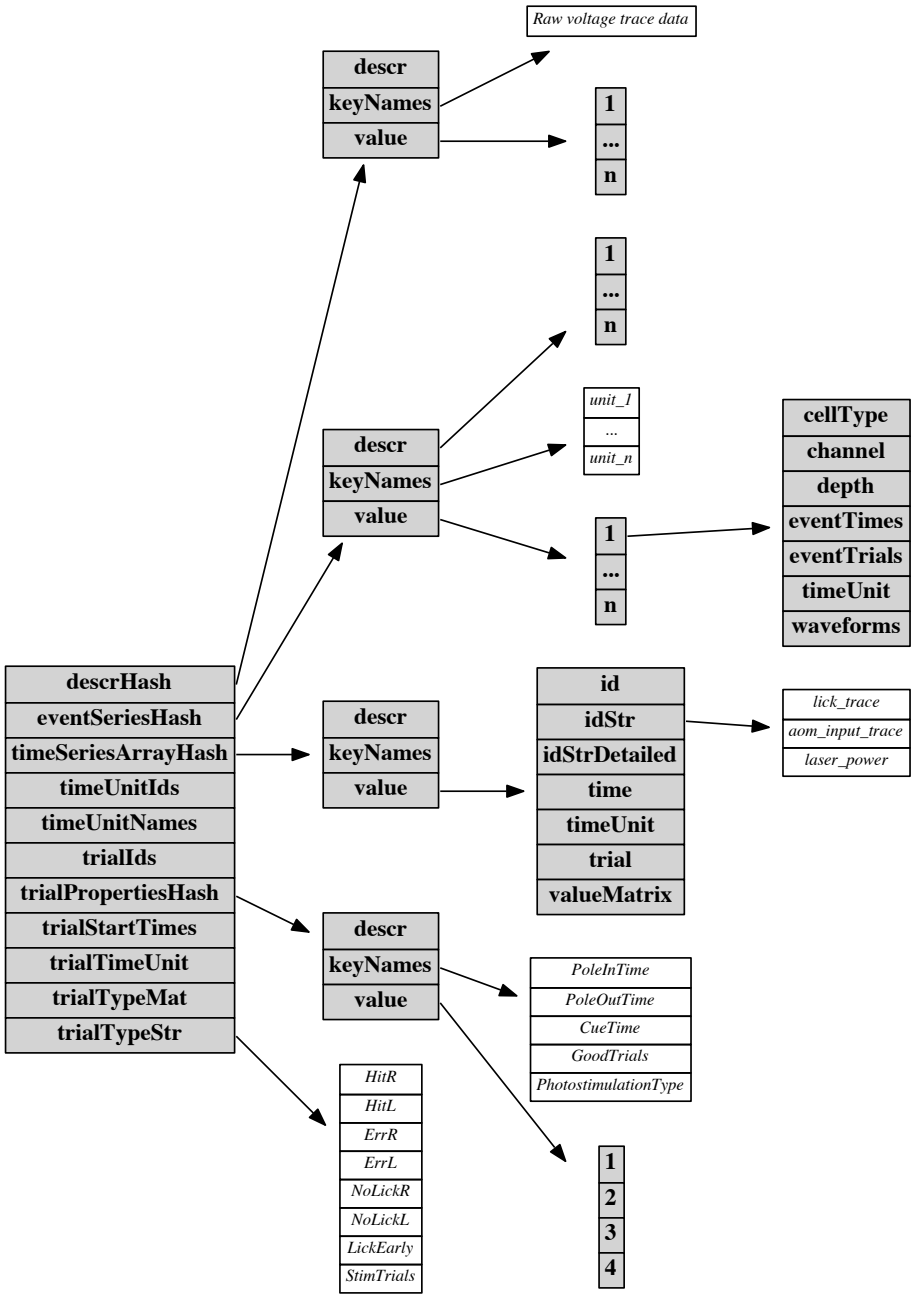


Figure 4. Structure of the CRCNS alm-1 data file. The graph shows the hierarchical relationship between the different data elements.

descrHash: contains the directory and filename of the raw data file
- **.keyNames**: type of data
- .**descr**: more detailed description of the entries
- **.value**: directory and filename of the entries

e**ventSeriesHash**: contains the spike times as well as neuron information
- **.keyNames**: name of the entry
- **.descr**: description of the entry
- **.value**: each entry contains the data structure from one neuron
  - **.(1…n)**
    - **.cellType**: {"pyramidal" or "FS" or "PT" or "IT"}
    - **.channel**: which channel on the silicon probe the neuron is recorded from.
    - **.depth**: estimated depth of the neurons, in micrometers
    - **.eventTimes**: spike times for all the events
    - **.eventTrials**: indicates which trial the spike times were from
    - **.timeUnit**: time unit of the data
    - **.waveforms**: snippets of spike waveform

timeSeriesArrayHash: contains the time series data for behavioral monitoring and photostimulation
- **.keyNames**: {"EphusVars"} from Ephus acquisition software
- **.descr**: describes the content of the data
- **.value**: contains the data for each trial
  - **.(1…n)**
    - **.id**: channel numbers in Ephus acquisition software
    - **.idStr**: description of the time series data
    - **.idStrDetailed**: more detailed description of idStr
    - **.time**: time stamps for the time series data
    - **.timeUnit**: time unit used
    - **.trial**: trial number for each sample in the time series,
    - **.valueMatrix**: time series data

timeUnitIds: A vector of integers, with the following convention: 1--ms; 2--second; 3--minute; 4--hour; 5—day

timeUnitNames: Description of time units

trialIds: trial number to reference to the trials

trialPropertiesHash: contains detailed information about trial structures and timing information
- **.keyNames**: {'PoleInTime' 'PoleOutTime' 'CueTime' 'GoodTrials' 'PhotostimulationType'}.
- **.descr**: describes entries in keyNames.
- **.value**: contains the values of the properties in keyNames for each trial

trialStartTimes: start times of the trials

trialTimeUnit: specifies the time unit of the data

trialTypeStr: description of the rows in trialTypeMat

trialTypeMat: each column describes one trial by the description in trialTypeMat

Figure 5. Description of the CRCNS alm-1 data file contents. This graph provides details for the different data elements.

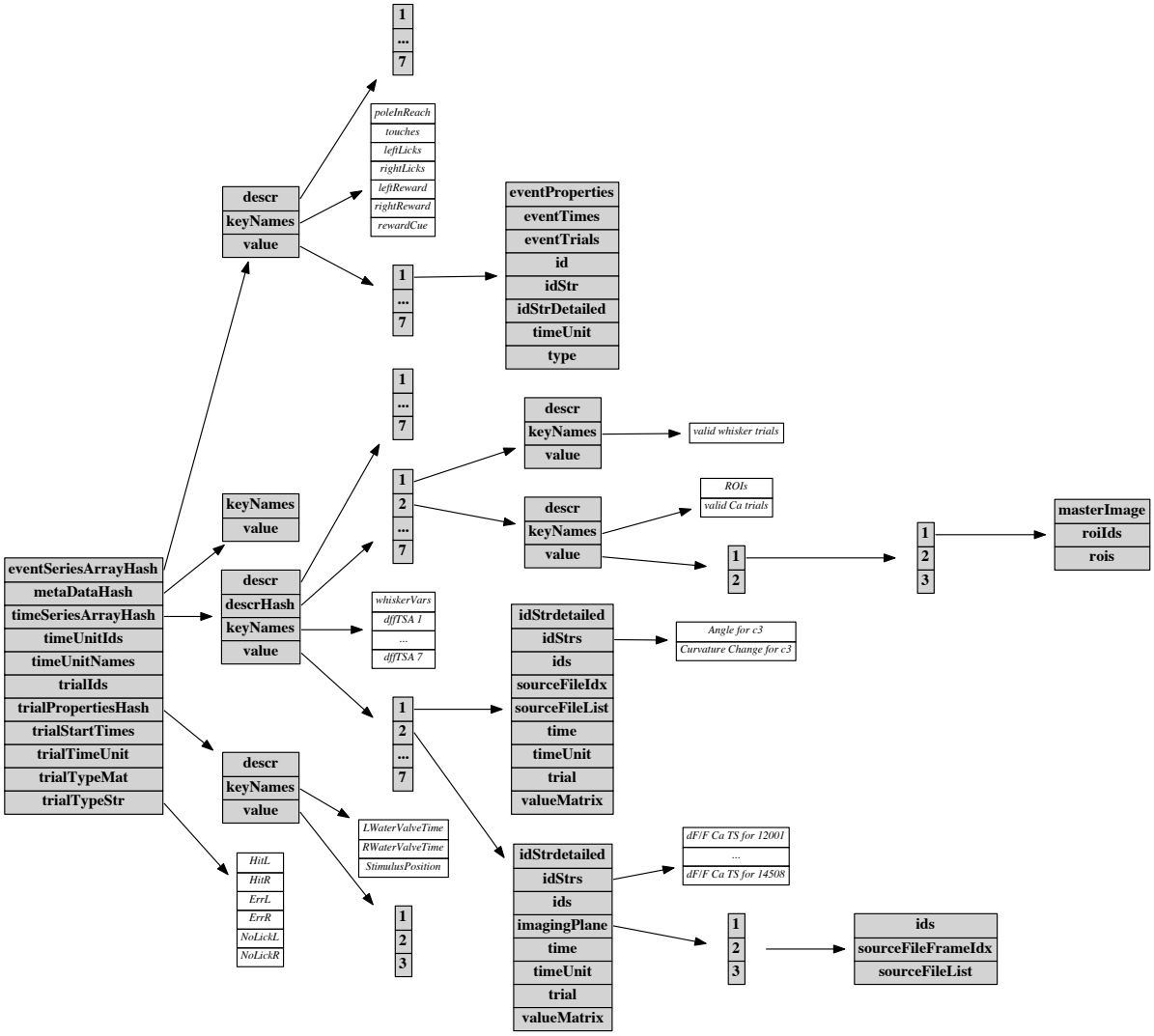


Figure 6. Structure of the CRCNS ssc-1 data file. The graph shows the hierarchical relationship between the different data elements.

eventSeriesArrayHash: contains any event series
- **.keyNames:** name of the entry
- **.descr:** description of the entry
- **.value:** each entry contains data in a structure
  - **.(1…n)**
    - **.eventProperties:** special event attributes for a given event
    - **.eventTimes:** times for all the events
    - **.eventTrials:** trial from which the event data were from
    - **.id:** numerical unique identifier for each data vector
    - **.idStr:** a description of what that particular vector is
    - **.idStrDetailed:** even more detailed description if needed
    - **.timeUnit:** what time unit is the data in
    - **.type:** 1 means events that only have a start ; 2 means events with start AND end

metaDataHash: Contains a series of fields describing each experiment

timeSeriesArrayHash: contains the time series
- **.keyNames:** what is stored in each hash
- **.descr:** more detailed description of what is in each hash
- **.value:** the actual timeseries data
  - **.(1…n)**
    - **.ids:** a numerical unique identifier for each data vector (ROI for imaging)
    - **.idStrs:** a description of what that particular vector is
    - **.idStrdetailed:** even more detailed description if needed
    - **.imagingPlane:**
      - **.(1…3)**
        - **.ids:** the ids of ROIs in this plane
        - **.sourceFileFrameIdx:** 2 x t vector, gives index of file name and frame for a given time point
        - **.sourceFileList:** the raw source data file
    - **.time:** time basis for this data
    - **.timeUnit:** the time unit used for this data
    - **.trial:** specifies the trial to which every time point belongs to
    - .**valueMatrix:** the data ; n x t matrix (t = time)
- **.descrHash:**
  - **.1:** stores a list of all trials for which whisker video was obtained
  - **.(2…n):** structure giving details for the calcium data
    - **.keyNames:** {ROIs, valid Ca data}
    - **.descr:** description of the keyNames
    - **.value:** data for .keyNames
      - **.1**
        - **.(1…3):** vector with IDs, that can index rois in imaging plane i.
          - **.masterImage:** reference image for this plane in which ROIs are drawn
          - **.roiIds:** the id of the ROI
          - **.rois:** an array with details for each ROI

timeUnitIds: A vector of integers, with the following convention: 1--ms; 2--second; 3--minute; 4--hour; 5—day

timeUnitNames: Description of time units

trialIds: trial number

trialPropertiesHash: information about trial structures and timing information
- **.keyNames:** brief description of fields
- **.descr:** describes entries in keyNames
- **.value:** contains the values of the properties in keyNames for each trial

trialStartTimes: start times of the trials

trialTimeUnit: specifies the time unit of the data

trialTypeStr: description of the rows in trialTypeMat

trialTypeMat: each column describes one trial by the description in trialTypeMat

Figure 7. Detailed description of the CRCNS ssc-1 data file contents. This graph provides details for the different data elements.

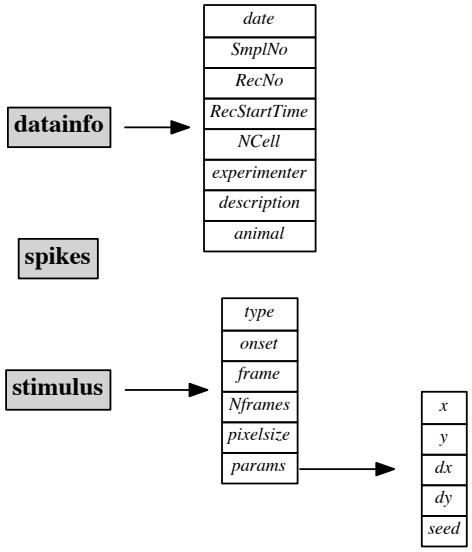


Figure 8. Structure of the .mat file. The graph shows the hierarchical relationship between the different data elements.

datainfo: a structure containing data information.
- **.date**: recording date (yyyymmdd)
- **.SmplNo**: retina number
- **.RecNo**: recording number
- **.RecStartTime**: recording start time stamp [Y, M, D, H, MN, S]
- **.Ncell**: number of simultaneously recorded cells
- **.experimenter**: experimenter name and affiliation
- **.description**: brief description of data
- **.animal**: animal genotype description

spikes: an M-by-N cell containing spike trains (in sec) of simultaneously recorded M cells in response to N stimuli.

stimulus: a 1-by-N structure with the following fields
- **.type**: stimulus type (e.g., binary white noise)
- **.onset**: stimulus onset time (in sec)
- **.frame**: frame length (in sec)
- **.Nframes**: total number of frames
- **.pixelsize**: pixel size on the retina (8.3 um/pixel)
- **.param**: stimulus-specific parameters, such as;
  - **.x**: total stimulus width (in pixels)
  - **.y**: total stimulus height (in pixels)
  - **.dx**: width of stimulus tiles (in pixels)
  - **.dy**: heigth of stimulus tiles (in pixels)
  - **.seed** : seed for random number generator Ran1

Figure 9. Detailed description of the CRCNS ret-1 data file contents. This graph provides details for the different data elements.

```
/kwik_version* [=2]
/name*
/application_data
    spikedetekt
        MY_SPIKEDETEKT_PARAM*
        ...
/user_data
/channel_groups
    [X]  # Absolute channel group index from 0 to Nchannelgroups-1
        name*
        channel_order*  # ordered list of channels, as specified in the PRB file
        adjacency_graph* [Kx2 array of integers]
        application_data
        user_data
        channels
            [X]  # Relative channel index from 0 to shanksize-1
                name*
                ignored*
                position* (a pair (x, y) in microns relative to the whole multishank
                voltage_gain* (a float32 number, in microvolts)
                display_threshold*
                application_data
                    klustaviewa
                    spikedetekt
                user_data
        spikes
            time_samples* [N-long EArray of UInt64]
            time_fractional* [N-long EArray of UInt8]
            recording* [N-long EArray of UInt16]
            clusters
                main* [N-long EArray of UInt32]
                original* [N-long EArray of UInt32]
            features_masks
                hdf5_path* ['{kwx}/channel_groups/X/features_masks']
            waveforms_raw
                hdf5_path* ['{kwx}/channel_groups/X/waveforms_raw']
            waveforms_filtered
                hdf5_path* ['{kwx}/channel_groups/X/waveforms_filtered']
        clusters
            [clustering_name]
                [X]  # Cluster number from 0 to Nclusters-1 (unique within a given c
                    application_data
                        klustaviewa
```

Figure 10. Excerpt of a KWIK file (from Klusta-team/kwiklib, 2015). The Graph shows part of the file hierarchy of a KWIK file.

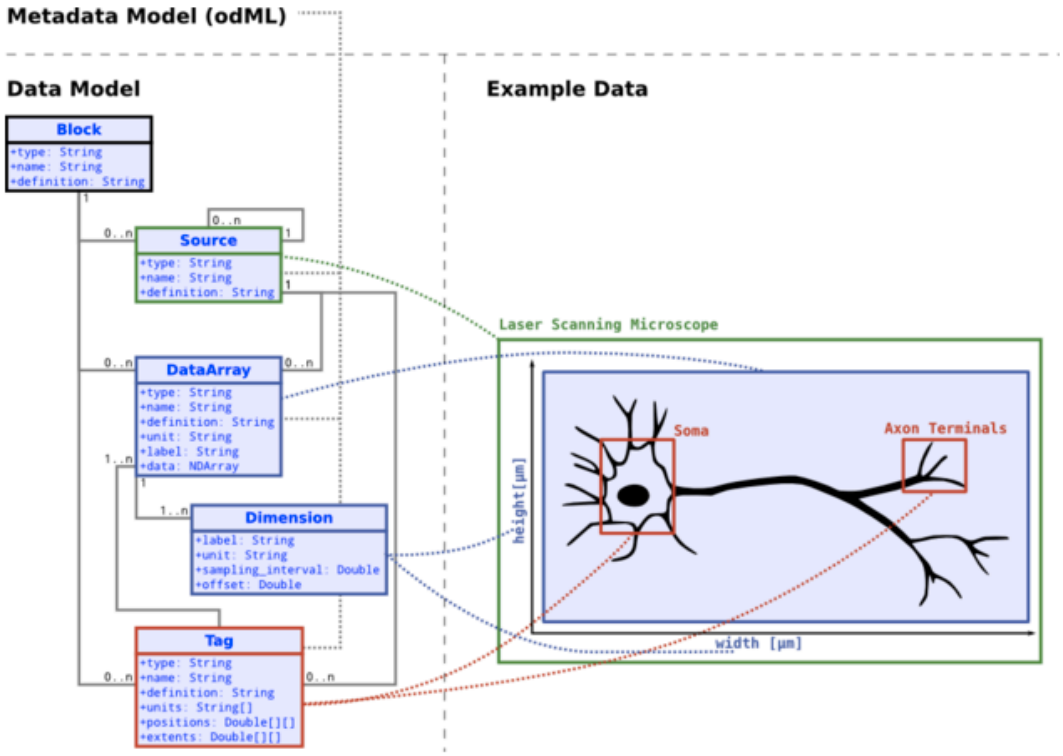


Figure 11. The Nix data model (from: G-Node/nix, 2015). The Graph shows the individual parts of the Nix data model and their connections with each other and the data.

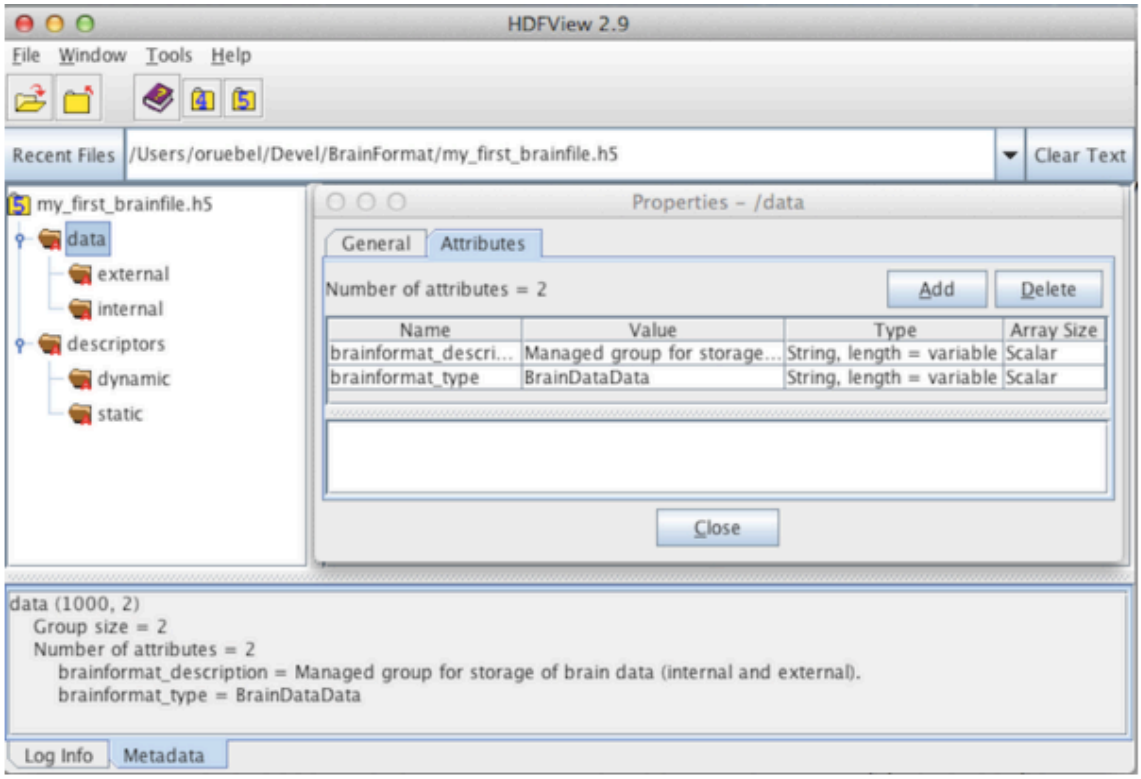


Figure 12. Basic Structure of a Brain File (from Brain Format, 2015). The graph shows the basic data hierarchy of a Brain file.

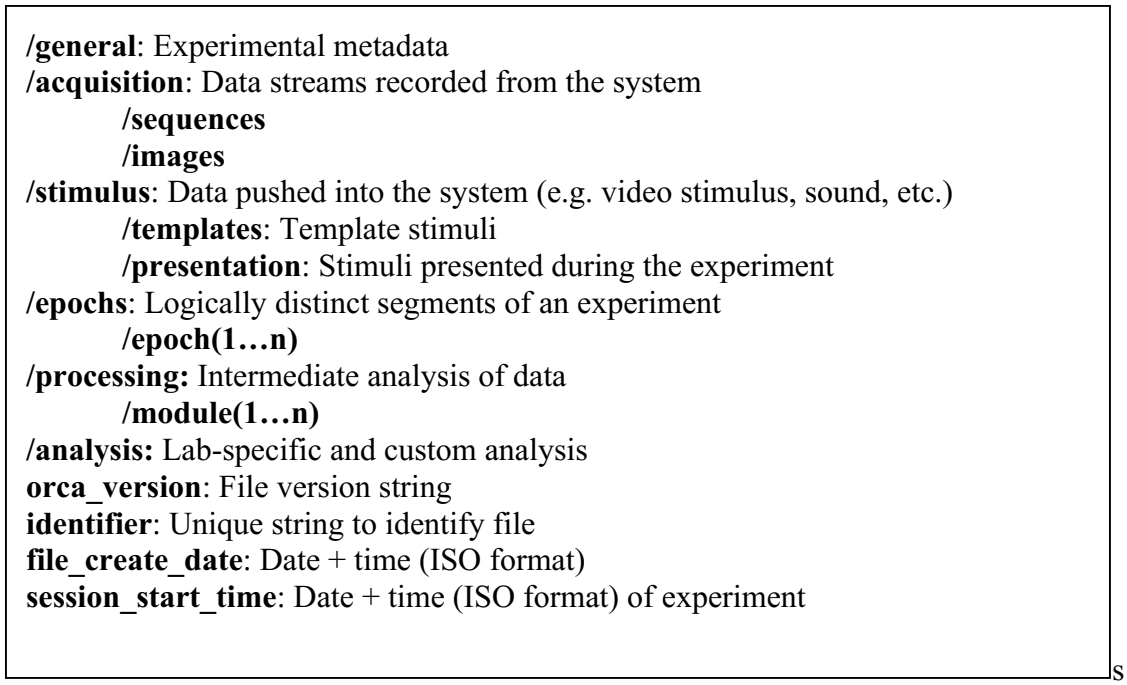


Figure 13. Basic file structure of an Orca file. Top-level groups of the Orca file are explained in this Figure.

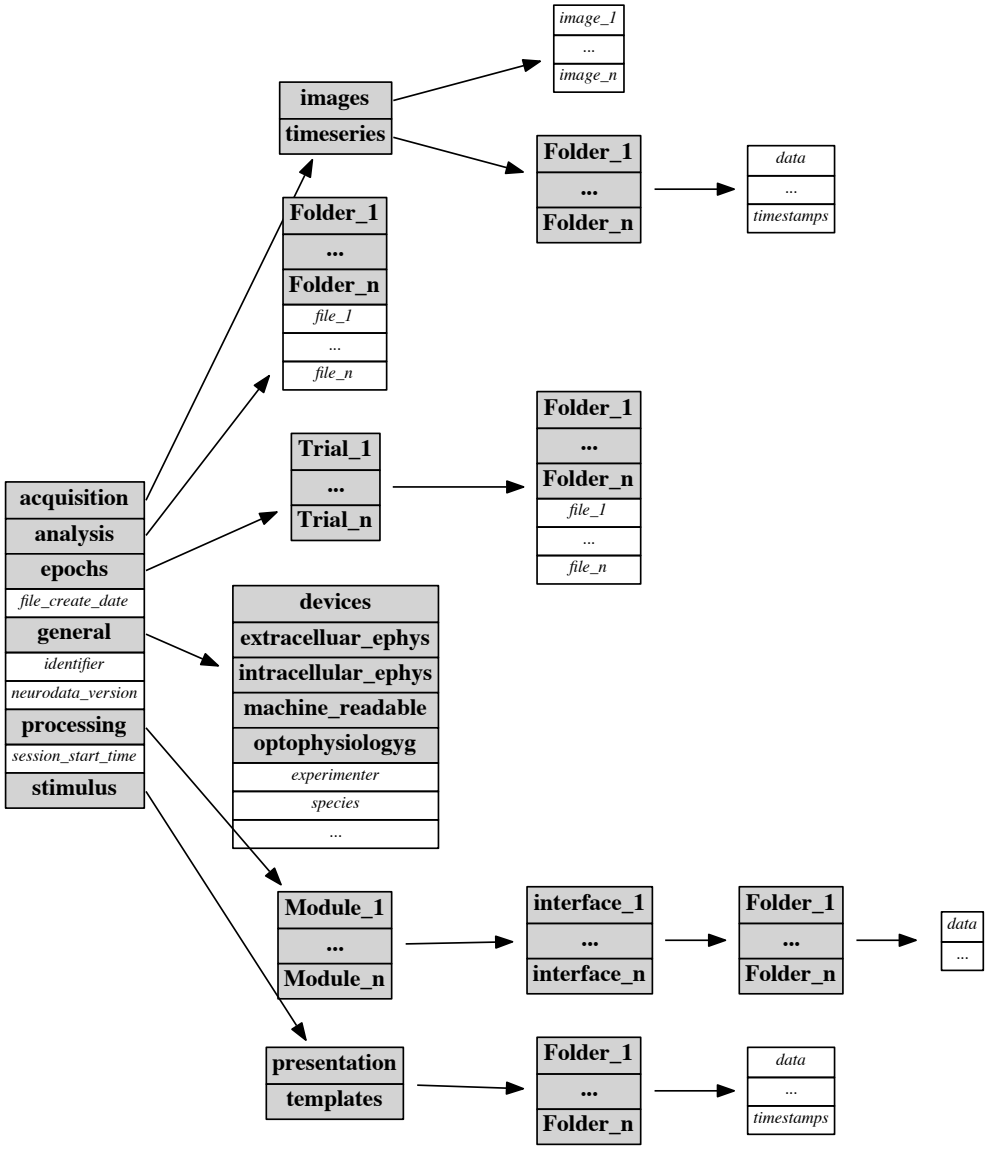


Figure 14. File structure of a Borg file. The graph shows the hierarchical relationship between the different data elements.

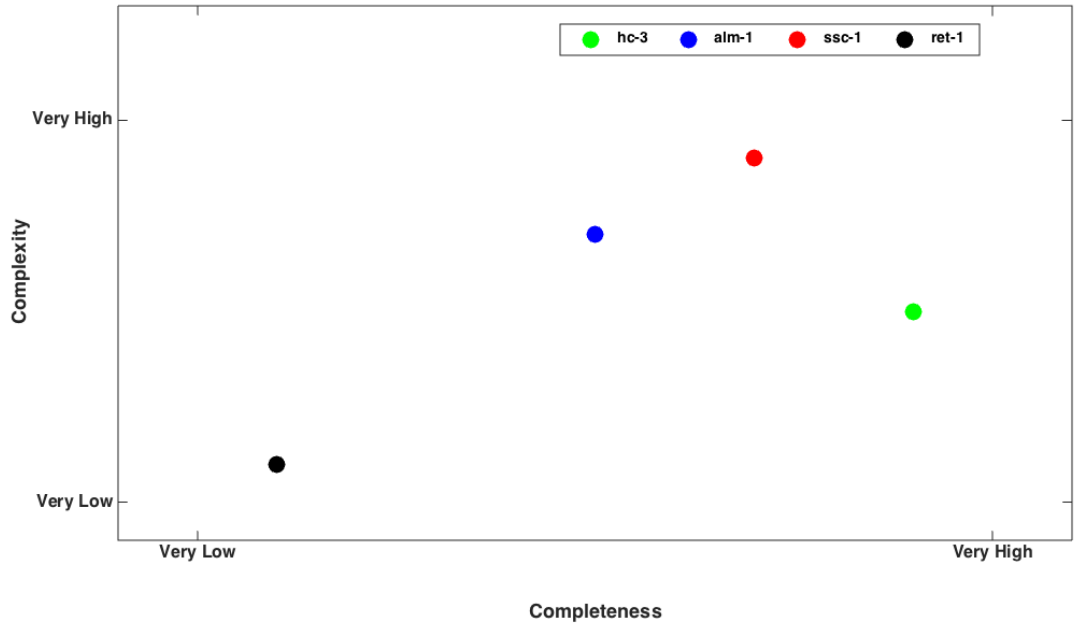


Figure 15. Complexity and Completeness of the Datasets. The Graph illustrates the degree complexity and the level of completeness of the different datasets in terms of raw and processed data.

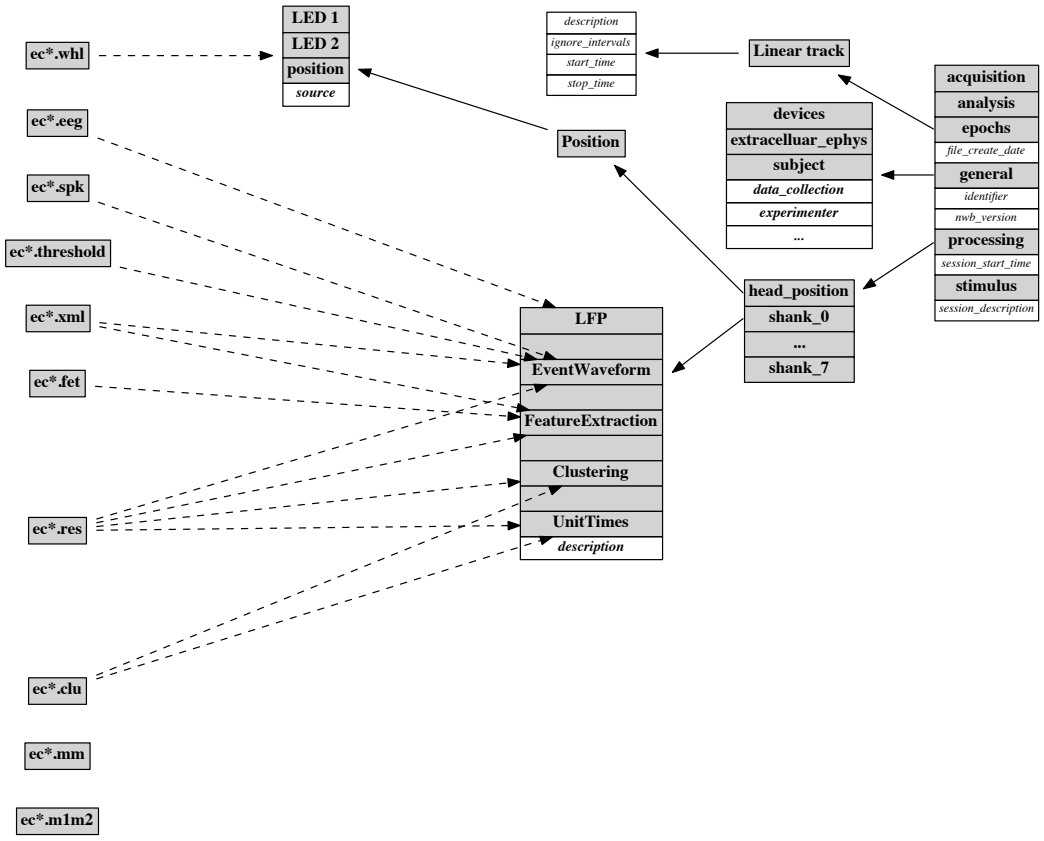


Figure 16. Mapping of CRCNS hc-3 onto NWB. The mapping results in a file with an unlike more complex file with a structured hierarchy.

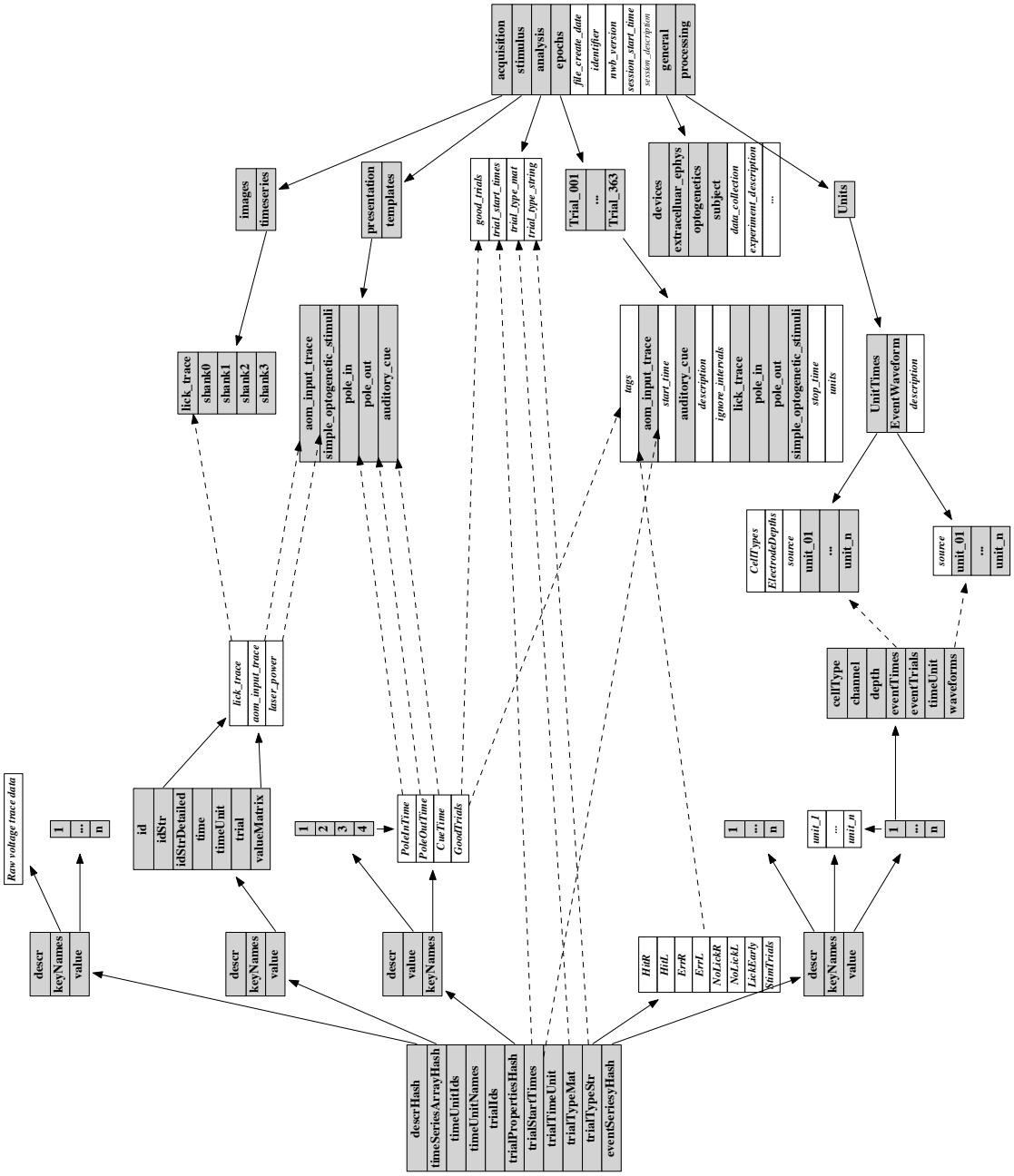


Figure 17. Mapping of CRCNS alm-1 onto NWB. The resulting file is of comparable complexity but has a different structure.

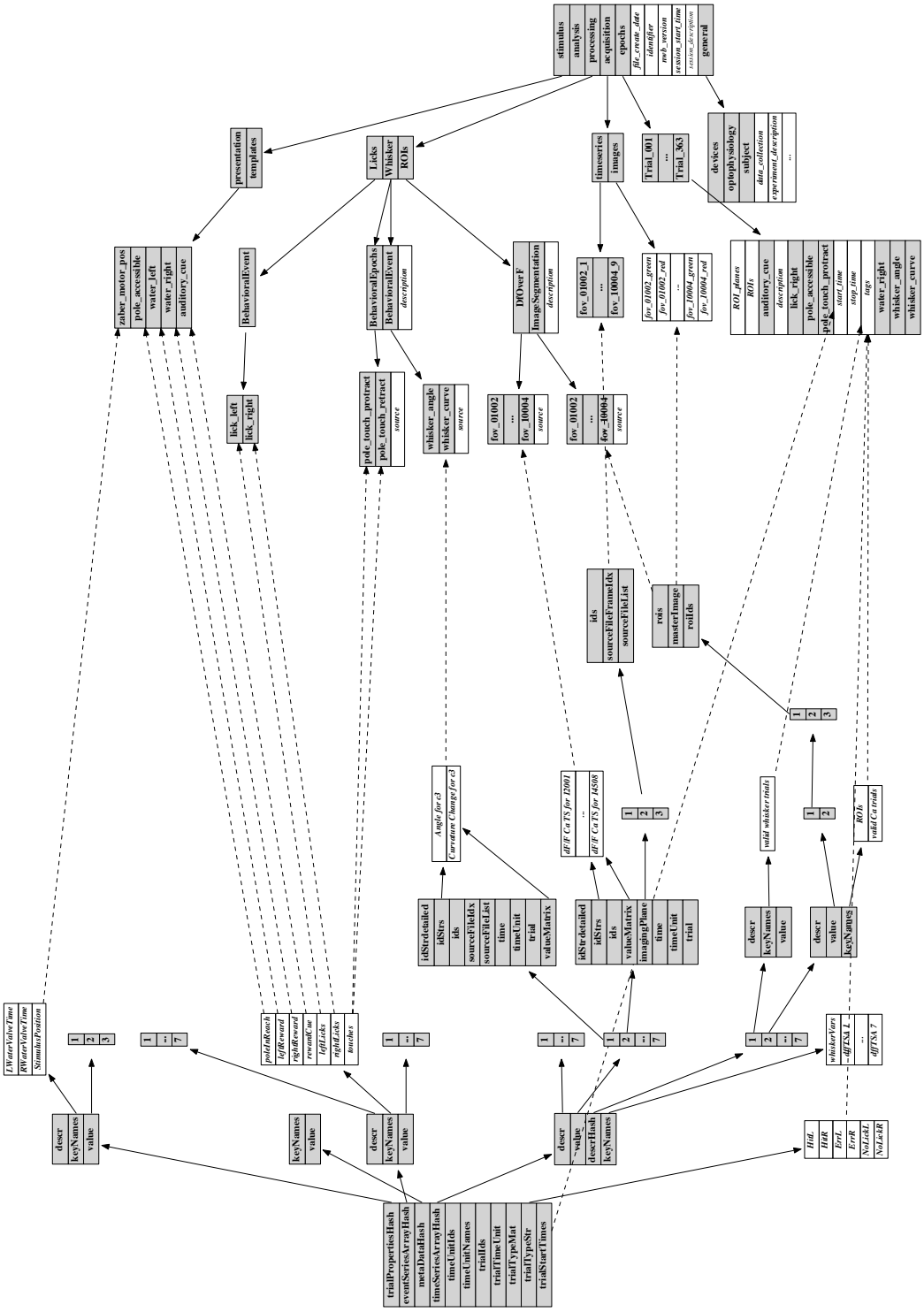


Figure 18. Mapping of CRCNS ssc-1 onto NWB. The resulting file is of comparable complexity but has a different structure.

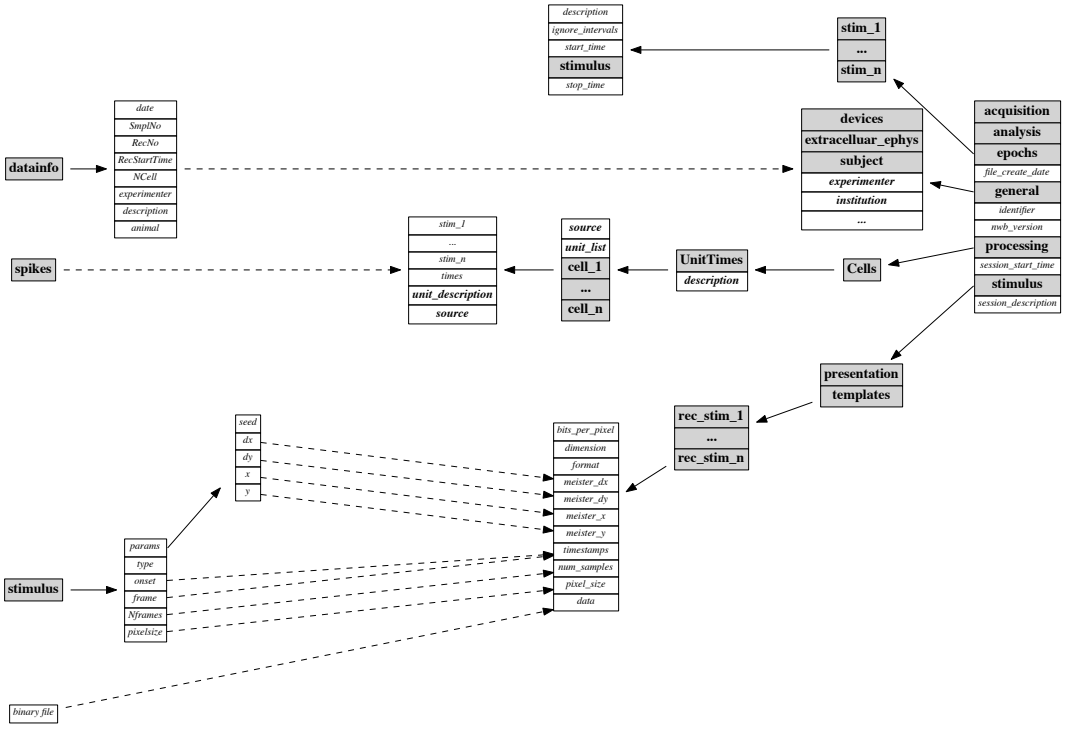


Figure 19. Mapping of CRCNS ret-1 onto NWB. The resulting file is unlike more complex.

Appendix II

Tables

Table 1. Model Data Sets (adapted from: NWB_hackathon1.pdf 2015). Overview of the data sets used to develop, test and evaluate the NWB file format.

Dataset Identifier	Origin	Description
CRCNS hc-3	György Buzsáki	Rat hippocampus: Multi-unit recordings from different rat hippocampal regions while the animals were performing several behavioral tasks
CRCNS pvc-6	Allen Institute	Mouse slice electrophysiology: In vitro intracellular recording and staining of a single neuron in the visual cortex of a mouse
CRCNS alm-1	Karel Svoboda	Electrophysiological recordings from rat barrel cortex: Extracellular recordings from ALM neurons of adult mice performing a tactile decision behavior
CRCNS ssc-1	Karel Svoboda	Calcium imaging of rat somatosensory cortex: Calcium imaging data from vibrissal S1 in mice performing a pole localization task
CRCNS ret-1	Markus Meister	Retina: In vivo calcium imaging of layer 4 cells in the mouse using sinusoidal grating stimuli.

Table 2. Evaluation Criteria identified by the NWB workgroup (adapted from: NWB_hackathon1.pdf 2015). The table list and explains the different evaluation criteria and specifies the type of metric for each criterium.

Criteria	Test	Metric
Ability to store/ represent EPhys and OPhys data	File format can store data from representative data sets on CRCNS	Pass/Fail
Open source	Project compatible license and dependencies are approved by the OSI	Pass/Fail
Extensibility	Data format can be extended to accommodate new methods, details, and features	Pass/Fail
Robust tool ecosystem	File format is compatible with being read-from/written -to by widely used software tools (eg, klustakwik for ephys), assuming modification to said tools	Pass/Fail
Python support	Data can be accessed using Python	Pass/Fail
Matlab support	Data can be accessed using Matlab	Pass/Fail
Java support	Data can be accessed using Java	Pass/Fail
C/C++ support	Data can be accessed using C or C++	Pass/Fail
View data without coding	One or more free apps to view the file's contents	Number of apps
Ease of importing new data sets	Time and resources required to import new data into the file forma	Benchmark time to perform this task for datasets of variable complexity
High read bandwidth for specific data	Read one image frame or one trace from the data file.	Benchmark averaged time to read
High read bandwidth for session	Read multiple image frames or traces from the data file.	Benchmark averaged time to read session
Storage	File size	Measure storage requirements

Table 3. Conversion of the evaluation criteria into numeric values. The table list the metric measures for the different criteria and explains the meaning of the assigned numbers.

Criteria	Measure	Comments
Ability to store/ represent EPhys and OPhys data	0-10	0: no data are stored/represented properly, 10: all example data sets are stored/represented, 1-4: partial storage in %
Open source	0-1	0: No, 1: Yes
Extensibility	0-5	0: no option for extension, 5: supports diverse extensions with ease, 1-4: intermediates in proportion measured in number of found limitations
View data without coding	0-1	0: No, 1: Yes
Python support	0-2	0: no Python support, 2: read and write API, 1: at least write API
Matlab support	0-2	0: no Matlab support, 2: read and write API, 1: at least write API
Robust tool ecosystem	0-5	0.0 if no third party software tool is supported, 5 if all major tools are supported, maximum has to be defined, 1-4 measure proportions of the maximum
Ease of importing new data sets	0-5	0 if it takes a developer >2 weeks, 5 if an average researcher needs 2 days or less, 1-4 measure in between these extremes
Additional apps for viewing	0-2	0.0 for no apps, 0.1 for 1-2, 0.2 for 3 or more
Java support	0-2	0: no Java support, 2: read and write API, 1: at least write API
C/C++ support	0-2	0: no C/C++ support, 2: read and write API, 1: at least write API
High read bandwidth for specific data	0-10	Number relates to the time it takes to access individual parts of the file format. 10 means that there is now noticeable difference after file conversion, 0 if there are significant wait times for the major parts of the data (>100%), 1-8 measure fractions between 100% delays and none
High read bandwidth for session	0-10	Benchmark averaged time to read session for the new file format compared to the old one. If this proportion is exceeding 2, 0 points are given. If it is 1, 10 points are given. 1-9 points measure intermediates between 2 and 1.
Storage	0-10	Number relates to new file size/original file size. If this proportion is exceeding 2, 0 points are given, if it is 1, 10 points are given. 1-9 measure intermediate proportions between 2 and 1.

Table 4. Mapping of the datasets on the modules of the 'what' document. The table gives an overview about which modules of the ‘what’ documents are contained in the data sets.

Module/Dataset	hc-3	alm-1	ssc-1	ret-1
Exp and Animal Information	x	x	x	x
Intracellular Electrophysiology				
Extracellular Electrophysiology	x	x		x
Optophysiology			x	
Sensory Stimuli		x	x	x
Simple Optogenetic Stimuli		x		
Behavioral Events	x	x	x	
Pharmacology				

Table 5. Key Features and their representation in the individual file formats. ‘-‘ means that a feature is not present, ‘+’ means that it exists in the individual file format, multiple ‘+” are used to differentiate file formats from each other with respect to a specific feature.

Feature/Format	NWB	Kwik	SLDF	Nix	Brain
Support wide range of data	++	-	+	+++	-
Support modular approach	+	+	+	+	+
User friendly, easy to understand	+	+	-	+	+
Provenance information	+	-	-	+	-
Define and retrieve sub-sections of data	++	-	+	++	+++
Support KlustaSuite	-	+	-	-	-
Meta data solution	++	+	+	+++	++
Use of human readable specification	+	-	-	-	+
Ease of developing/handling API for outsiders	++	-	-	-	+
Multitude of supported languages	++	-	-	+++	-
File validation tools	+	-	-	+++	+++
Documentation and Tutorials	+	++	+	++	++

Table 6. Scorecard for the Borg format. The table lists the scores that the Borg format achieved for the different evaluation criteria.

Criteria	Measure	Score
1. Ability to store/ represent EPhys and OPhys data	0-10	10
2. Open source	0-1	1
3. Extensibility	0-5	4
4. View data without coding	0-1	1
5. Python support	0-2	1
6. Matlab support	0-2	0
7. Robust tool ecosystem	0-5	0
8. Ease of importing new data sets	0-5	2
9. Additional apps for viewing	0-2	1
10. Java support	0-2	0
11. C/C++ support	0-2	0
12. High read bandwidth for specific data	0-10	4
13. High read bandwidth for session	0-10	2.75
14. Storage	0-10	2.5

Table 7. Performance comparison for loading single datasets into Matlab. The table compares the times required to open single datasets in the Borg files with the times required to open the same datasets in the original files. The ratios and the resulting scores are listed in the last two lines.

	hc-3	alm-1	ssc-1	ret-1
Original File	0.0856 s	0.1978 s	0.0002 s	0.3800 s
Borg File	0.4255 s	0.2699 s	0.0021 s	0.2279 s
Ratio Borg/Original	4.97	1.36	10.5	0.60
Score	0	6	0	10

Table 8. Performance comparison for running different Matlab scripts. The table compares the times required to perform certain session scripts for both the file in Borg format and the original files. The ratios and the resulting scores are listed in the last two lines.

	Script_1	Script_2	Script_3	Script_4
Original File	0.0032 s	0.0027 s	0.0144 s	6.4485 s
Borg File	0.0062 s	0.0459 s	0.0326 s	5.1692 s
Ratio Borg/Original	1.94	17	2.26	0.80
Score	1	0	0	10

Table 9. Storage requirements for the original and the Borg files. The table lists and compares the file sizes of the original and the Borg files. The ratios and the resulting scores are listed in the last two lines.

	hc-3	alm-1	ssc-1	ret-1
Original File	285 MB	165 MB	143 MB	39 MB
Borg File	282 MB	594 MB	456 MB	160 MB
Ratio Borg/Original	0.99	3.6	3.19	4.1
Score	10	0	0	0

Table 10. Scorecard for the NWB format. The table lists the scores that the NWB format achieved for the different evaluation criteria.

Criteria	Measure	Score
1. Ability to store/ represent EPhys and OPhys data	0-10	10
2. Open source	0-1	1
3. Extensibility	0-5	5
4. View data without coding	0-1	1
5. Python support	0-2	1
6. Matlab support	0-2	1
7. Robust tool ecosystem	0-5	0
8. Ease of importing new data sets	0-5	4
9. Additional apps for viewing	0-2	1
10. Java support	0-2	0
11. C/C++ support	0-2	0
12. High read bandwidth for specific data	0-10	3.5
13. High read bandwidth for session	0-10	4.25
14. Storage	0-10	6

Table 11. Performance comparison for loading single datasets into Matlab. The table compares the times required to open single datasets in the NWB files with the times required to open the same datasets in the original files. The ratios and the resulting scores are listed in the last two lines.

	hc-3	alm-1	ssc-1	ret-1
Original File	0.0856 s	0.1978 s	0.0002 s	0.3800 s
NWB File	0.4050 s	0.3140 s	0.0016 s	0.2251 s
Ratio NWB/Original	4.73	1.59	8	0.59
Score	0	4	0	10

Table 12. Performance comparison for running different Matlab scripts. The table compares the times required to perform certain session scripts for both the file in NWB format and the original files. The ratios and the resulting scores are listed in the last two lines.

	Script_1	Script_2	Script_3	Script_4
Original File	0.0032 s	0.0027 s	0.0144 s	6.4485 s
NWB File	0.0043 s	0.0450 s	0.0568 s	5.0791 s
Ratio NWB/Original	1.34	16.7	3.94	0.79
Score	7	0	0	10

Table 13. Storage requirements for the original and the NWB files. The table lists and compares the file sizes of the original and the NWB files. The ratios and the resulting scores are listed in the last two lines.

	hc-3	alm-1	ssc-1	ret-1
Original File	285 MB	165 MB	143 MB	39 MB
NWB File	283 MB	107 MB	240 MB	122 MB
Ratio NWB/Original	0.99	0.65	1.65	3.13
Score	10	10	4	0

Table 14. Stimuli represented in the Datasets. The table lists which stimuli are included in the different data sets.

	hc-3	alm-1	ssc-1	ret-1
Pole		x	x	
Auditory		x	x	
Water			x	
Visual				x

Table 15. Behavioral Events found in the Datasets. The table lists which Behavioral events are found in the different data sets.

	hc-3	alm-1	ssc-1	ret-1
Head Position	x			
Licks		x	x	
Pole Touch			x	
Whisker Position			x	

Table 16. Overview about all empty entities in all four NWB files. The table gives an overview about the empty fields found in the different data sets.

Empty field/folder	hc-3	alm-1	ssc-1	ret-1
/acquisition/images	X	X		X
/acquisition/timeseries	X			X
/analysis	X		X	X
/general/devices	X			X
/epochs/epoch_n/description	n/a	X		X
/epochs/epoch_n/ignore_intervals	n/a	X	X	X
/stimulus/presentation	X			
/stimulus/templates	X	X	X	X

Table 17. Overview about the custom fields in the NWB files. The table lists all the different custom fields found in the NWB files for all data sets.

Fieldname	Origin	No.
/processing/shank_0/EventWaveform/waveform_timeseries/sample_length	hc-3	1
/processing/Units/EventWaveform/unit_n/sample_length	alm-1	2
/epoch/Trial_n/units	alm-1	3
/processing/Units/UnitTimes/unit_n/trial_ids	alm-1	4
/epochs/Trial_n/ROI_planes	ssc-1	5
/epochs/Trial_n/ROIs	ssc-1	6
/processing/ROIs/DfoverF/fov_n/trial_ids	ssc-1	7
/processing/Units/UnitTimes/CellTypes	alm-1	8
/processing/Units/UnitTimes/ElectrodeDepths	alm-1	9
/processing/Whisker/BehavioralEpochs/pole_touch_*/kappa_max_abs_over_touch	ssc-1	10
/processing/Cells/UnitTimes/cell_n/stim_n	ret-1	11
/stimulus/presentation/meister_dx	ret-1	12
/stimulus/presentation/meister_dy	ret-1	13
/stimulus/presentation/meister_x	ret-1	14
/stimulus/presentation/meister_y	ret-1	15
/stimulus/presentation/pixel_size	ret-1	16

Appendix III

Glossary

“AIBS”: Allen Institue for Brain Science in Seattle.

“API”: Application Programming Interface. A set of routines, protocols, and tools for building software applications.

“BRAIN” initiative : Brain Research through Advancing Innovative Neurotechnologies. Initiative started in 2013 by President Barack Obama.

“CARMEN”: Code, Analysis, Repository and Modelling for e-Neuroscience. Neuroscience data sharing initiative started in 2006.

“CARMEN NDF”: Neuroscience Data Format. Developed by the CARMEN consortium in MATLAB file format.

“CRCNS”: Collaborative Research in Computational Neuroscience. Data sharing initiative, started in 2002.

“Data model”: Organizes data elements and defines how the data elements relate to one another.

“EDF”: European Data Format. File format for EEG data, developed in 2003.

“EEG”: Electroencephalography (EEG) is the recording of electrical activity by electrodes placed along the scalp.

“Electrophysiological data”: data, which describe the electrical properties of biological cells and tissues.

“GDF”: General Data Format. File format for EEG data, developed in 2006.

“G-Node”: German nodes of the INCF

“HDF5”: acronym for “Hierarchical Data Format” (version 5), architecture for a file format, optimized for big datasets.

“HBI”: Human Brain Initiative. European 10-year scientific research project, established in 2013.

“INCF”: International Neuroinformatics Coordinating Facility. International science organization, which aims at supporting cooperation in the neuroinformatics field.

“Metadata”: data about data. Used to describe details about experimental data.

“KWIK file format”: file format, used by the KlustaKwik spike sorting suite

“LBNL Brain format”: file format developed by the Lawrence Berkeley National Laboratory.

“Markup Language”: System for annotating a document. XML and HTML are prominent examples.

“MINI”: A set of minimal common metadata for neurodata. Developed by the CARMEN consortium.

“MEF format”: Multiscale Electrophysiology File format. Mainly used for EEG data.

“NEO API”: Python based successor of Neo. Developed by G-Node.

“Neuroinformatics”: Field of research, which applies computational models and analytical tools to describe neuroscience data.

“Neuron”: Electrically excitable cell, capable of transmitting and processing information through electrical and chemical signals.

“Neurophysiology”: Field of research that is concerned with the study of the functioning of the nervous system

“Neuroshare API”: prominent API for neurodata. Developed by G-Node

“NIX format”: C++ based file format, developed by G-Node.

“NWB”: Neurodata without borders. Initiative launched in 2014 with the goal to develop a common file format for neurophysiology data.

“object”: data structures that contain data, in the form of fields and behavior to operate on the encapsulated data.

“Object-oriented design”: is the process of designing a system of interacting objects for to address a software problem.

“odML”: Initiative to define and establish an open, flexible and easy-to-use format to transport metadata. Developed by G-Node.

“OECD”: Organization for Economic Co-operation and Development.

“OEN”: Ontology for Experimental Neurophysiology

“Ontology”: Organizational system designed to categorize and help explain the relationships between various concepts of science in the same area of knowledge and research.

“Optophysiological data:” Data obtained by optical method to study aspects of physiology.

“Polygraphic recordings”: Measurement and recording of several physiological indices .

“Python wrapper”: Collection of subroutines or classes to translates a library's existing interface into a Python compatible interface.

“Standardization”: Creating a common standard. In the context of file format it means one standard for all addressed data files.

“Provenance”: the ability to track derived data through each stage of the analysis

“Voltage-clamp”: Experimental method used by electrophysiologists to measure ion current

Appendix IV

Dataset Descriptions from Hackathon I Handout (NWB_hackathon1.pdf 2015)

CRCNS alm-1

LAB: Janelia

SUMMARY: Extracellular recordings from ALM neurons of adult mice performing a tactile decision behavior

METHODS:
- Extracellular recording
- Tactile stimulus and auditory cue
- Optogenetics

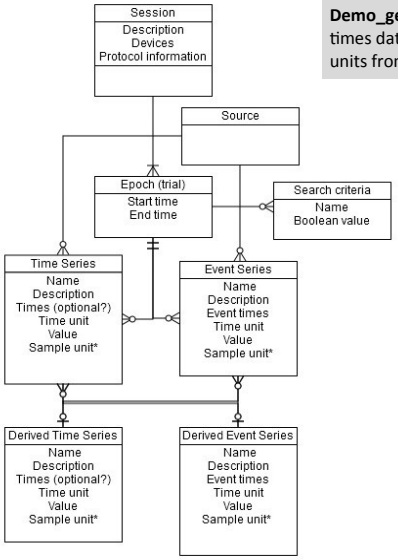
DETAILS:
- Species
- Genotype
- Surgical procedures
- Recording equipment
- Recording depth
- Spike sorting methods
- Tactile stimulus parameters
- Protocol

ELECTRICAL:
- Voltage trace
- Lick trace

OPTICAL:

STIMULI:
- Pole movement
- Auditory cue
- Photostimuluation

TRACKING:
- Spike sorting
- Lick events
- Neuron type classifiation



ANALYSIS GOALS

These experiments probe neural dynamics in the anterior motor cortex (ALM). They simultaneously record the electrical activity and optogenetically stimulate neurons in multiple areas of the anterior motor cortex (ALM) to understand the ALM's relationship to voluntary movement.

Primary analysis detects and sorts spikes, then combines these data with neuron classification and behavioral output.

1. Extracellular recording traces were band-pass filtered and some manual spike sorting was performed to extract clear single units. Spikes are clustered into units based on waveform shape. Each unit is suspected to be a single neuron. Spikes were manually sorted, and units manually categorized. spike-width of <35 ms: putative fast-spiking neurons, spike-width of >45ms: putative pyramidal neurons, units with intermediate values were excluded from analysis
2. Neurons were classified combining photo stimulation and spike train data. Presorted units were categorized as either **Pyramidal-tract (PT)** neurons or **Intratelencephalic** (IT) neurons with antidromic photostimulation and a collision test which looked for absence of antidromic spikes when they were preceded by spontaneous spikes
3. Behavioral events are extracted and all time stamps aligned

Downstream analysis:
- Calculate PSTH for a single unit
- Calculate Cross correlation between units
- Calculate Correlation between units and any behavioral variable

Example Scripts:

Demo_get_performance: this script extracts behavioral data from session object and plots the performance

Demo_get_trial_aligned_raster_PSTH: this script extracts spike times data from session object, it plots the raster and PSTH for all units from the session

1

CRCNS hc-3

LAB: NYU

SUMMARY: Multi-unit recordings from different rat hippocampal regions while the animals were performing several behavioral tasks

METHODS:
- Extracellular recording
- Behavioral task stimulus

DETAILS:
- Animal information
- Cell information
- Session information (behavior, familiarity, etc)
- Surgical procedures
- Spike sorting method

ELECTRICAL:
- Voltage trace
- EEG

OPTICAL:
- Voltage trace
- EEG

STIMULI:

TRACKING:
- Animal position

ANALYSIS GOALS

Investigate the role of self-organized neural rhythms in processing and segmenting stimuli and generating response.

In order to analyze the role of rhythms, broadband neural signals are recorded from behaving animals from multiple electrode "shanks" each with multiple recording sites. Each recording session has a single animal and recording electrode location/configuration. These recordings are processed to identify the activity of individual neurons. For each shank:

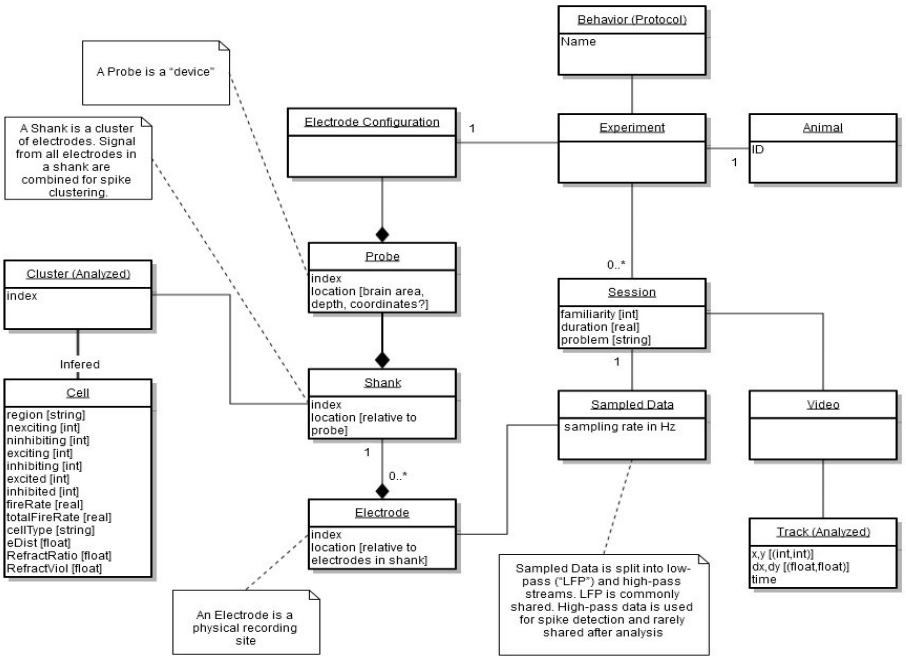
Neural signals are high-pass filtered and thresholded to identify putative spike times. Threshold crossing on any electrode is sufficient to record a "spike".

- Waveforms surrounding the spike time are collected from all electrodes in the shank containing the electrode that registered the "spike" in (1).
- Dimensionality reduction (PCA) is performed on collected waveforms
- Automatic clustering (KlustaKwik; http://klusta-team.github.io/) is performed on extracted spike features
- Manual adjustment of clusters is performed (Klusters)
- Spikes from identified clusters are re-sorted into sessions
- Clusters (i.e. cells) are analyzed for intrisic and network interaction properties

Simultaneous video is recorded during some behaviors. In these cases:

- Animal head location and orientation are calculated from the video

Combined cell activity is correlated with animal behavior (video tracking)



2

CRCNS pvc-6

LAB: Allen Institute

SUMMARY: In vitro intracellular recording and staining of a single neuron in the visual cortex of a mouse

METHODS:
- Intra-cellular current injection
- Intra-cellular staining

DETAILS:
- Species
- Sex
- Age
- Genotype
- Recording location(s)
- Electrode
- Slice preparation
- Amplifier
- Software
- Sampling rate
- Resting membrane potential
- QC waves

ELECTRICAL:
- Voltage

OPTICAL:

STIMULI:
- Current injection

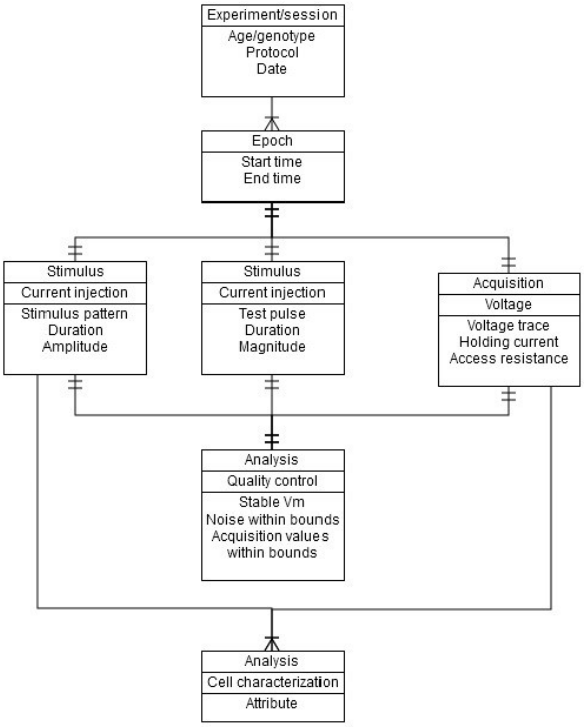
TRACKING:

ANALYSIS GOALS

Investigate whether mouse lines Contribute to cell classification efforts by studying diversity in neuronal cell classes and correlating the intrinsic excitability of neuron cell classes with their morphology and transcriptional information.

- Perform first order feature extraction including extracting amplitude of input stimuli and corresponding action potential (including height, width, slope, rate of rise, etc)
- Perform second order feature extraction including averaging across sweeps and sets

Cluster analysis using combination of first order and second order feature extraction and binning according to cell location feature clusters



CRCNS ret-1

LAB: CalTech

SUMMARY: In vivo calcium imaging of layer 4 cells in the mouse using sinusoidal grating stimuli.

METHODS:
- 2-photon imaging
- Visual stimulus

DETAILS:
- Species
- Sex
- Age
- Genotype
- Recording location
- Surgical procedure(s)
- Software
- Behavior box
- Microscope

ELECTRICAL:

OPTICAL:
- Image collection frequency
- Image resolution

STIMULI:
- Drifting sinusoidal gratings

TRACKING:
- Eye tracking

ANALYSIS GOALS

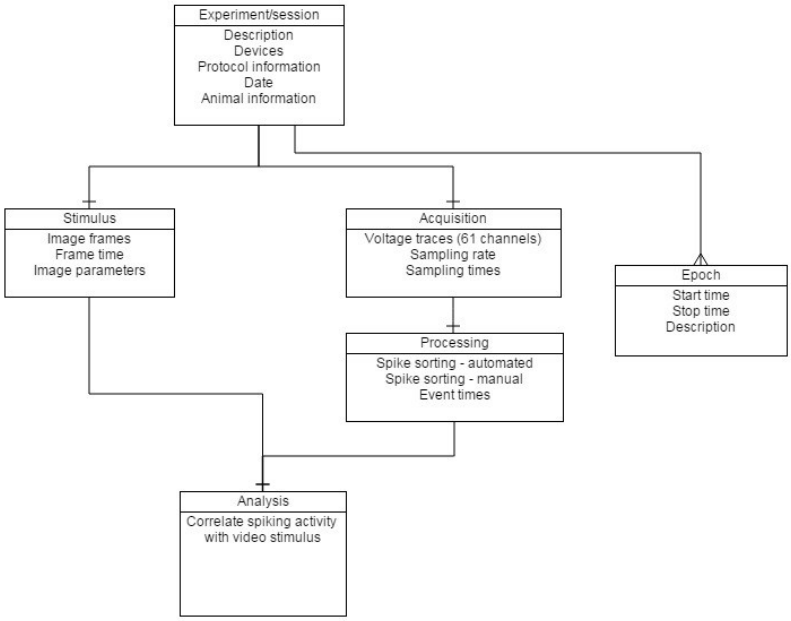
These data were collected for comparing visual response properties of retinal ganglion cells between wild-type and mutant mice. The following properties were examined in Lefebvre, et al. (2008):

- Size of the receptive field center (measured as the full width at half maximum of the receptive field profile)
- Speed of the response (measured as the time to peak of the temporal integration function)
- Average firing rate (observed during stimulation with flickering gratings)
- Threshold and gain of responses (in a linear-nonlinear model).

Reverse correlation (spike-triggered average) methods were used to recover the receptive fields of individual neurons.

The data can also be useful for analyzing the following properties:

- Physiological classification of neuron types
- Modeling neuron's input-output function
- Firing pattern correlation among neurons



6

CRCNS ssc-1

LAB: Janelia

SUMMARY: Calcium imaging data from vibrissal S1 in mice performing a pole localization task

METHODS:
- 2-photon imaging
- Tactile stimulus
- Auditory cue

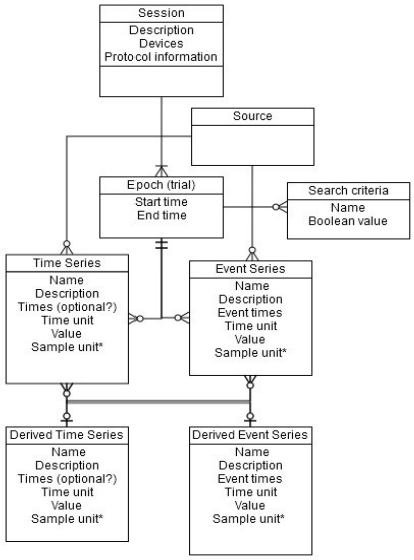
DETAILS:
- Species
- Genotype
- Surgical procedures
- Microscope
- Recording location
- Tactile stimulus parameters
- Protocol

ELECTRICAL:
- Lick trace

OPTICAL:
- Position of each image plane
- Image collection frequency
- Image resolution

STIMULI:
- Pole movement
- Auditory cue

TRACKING:
- Eye tracking
- Whisker angle
- Whisker curvature



ANALYSIS GOALS

This dataset comprises calcium imaging data from vibrissal S1 in mice performing a pole localization task.

1. Whisker video was collected at 500 Hz. The identity of whiskers across frames was tracked using a custom, automated MATLAB package.
- Whisker angle (at the base) and curvature were measured. Forces on the whiskers were presented as change in curvature Dk. Whisker curvature was measured a fixed distance range along the fitted polynomial.
- Periods of contact between whisker and object were detected based on the nearest distance between whisker and object and Dk. Whisker touches were first detected using an automated algorithm, then verified manually.

2. Imaging data are processed in several steps.
- First, image registration is performed to correct for brain movement by applying a line-by-line correction algorithm (based on a correlation based error metric).
- Registered data are aligned between imaging days.
- Regions of interest (ROIs) are drawn based on neuronal shape. Mean, maximum intensity and standard deviation were used to determine the boundaries of the neurons, and an automated method was used to align the ROIs across sessions.
- Raw fluorescence is extracted for each ROI. Neuropil subtraction is performed, and baseline fluorescence is estimated for each ROI, allowing for the calculation of DF/F. Finally, the DF/F trace is fit algorithmically to extract individual calcium events. To produce an event vector a non-negative deconvolution method was used.

3. Behavorial features were decoded based on activity.
- Machine learning algorithms (random forest algorithm, TreeBagger class in Matlab) were used for decoding
- Pearson correlation between model and estimated data was used to evaluate the accuracy of decoding.

4. Classification of the response type
- R2 was measured between each measured behavioral variable and each cell's decoder prediction
- ANOVA was used to determine whether contact-evoked calcium response was different for different pole positions

Example Script:

plotTrial: plots a single trial with relevant behavioral variables .

Appendix V:

Borg File Format Documentation

Borg – the NEUrophysiology Format

The Borg format assimilates data from many sources, including Ken and Karel's format proposed at the first hackathon; the Allen Institute's internal format; comments/requests by Ken, Karel and Marcus; characteristics of the datasets provided by collaborators; as well as features from other file formats.

File organization – Top-level groups (folders):

Group	Description	Comment
/general	Experimental metadata, including protocol, notes and description of hardware device(s)	Metadata representation follows the structure that was laid out in the hackathon proposal.
/acquisition	Data streams recorded from the system, including ephys, ophys, tracking, etc.	This folder is read-only after experiment complete and timestamps are corrected to common timebase. The data stored here may be links to raw data stored in external HDF5 files. This will allow keeping bulky raw data out of the file while preserving the option of keeping some/all in the file.
/stimulus	Data pushed into the system (eg, video stimulus, sound, voltage, etc)	This folder is read-only after experiment complete and timestamps are corrected to common timebase. Stores both presented stimuli and stimulus templates, the latter in case the same stimulus is presented multiple times, or is pulled from an external stimulus library.
/epochs	Experimental intervals, whether that be logically distinct sub-experiments having a particular scientific goal, trials during an experiment, or epochs deriving from analysis of data.	Epochs provide pointers to time series that are relevant to the epoch, and windows into the data in those time series (i.e., the start and end indices of *TimeSeries::data[ ]* that overlap with the epoch). This allows easy access to a range of data in specific experimental intervals.
/processing	The home for processing *Modules*. These modules perform intermediate analysis of data that is necessary to perform before scientific analysis. Examples include spike clustering, extracting position from tracking data, stitching together image slices.	*Modules* are defined below. They can be large and express many data sets from relatively complex analysis (e.g., spike detection and clustering) or small, representing extraction of position information from tracking video, or even binary "lick/no-lick" decisions. Common software tools (e.g., klustakwik, MClust) are expected to read/write data here.
/analysis	Lab-specific and custom scientific analysis of data. There is no defined format for the content of this folder – the format is up to the individual user/lab.	To facilitate sharing analysis data between labs, the contents here should be stored in standard types (eg, INCF types) and appropriately documented.

The contents of these organizational groups is more fully described on page 9. The Borg format is based on *TimeSeries* and *Modules* and these are defined first. In this document, the HDF5 terms "group" is often referred to as a "folder".

Borg stores general optical and electrical physiology data in a way that should be understandable to a naive user after a few minutes using looking at the file in an HDF5 browser. The format is designed to be friendly to and usable by software tools and analysis scripts, and to impose as few a priori assumptions about data representation and analysis as is deemed practical. The format include machine-readable metadata (e.g., units) necessary for analysis tools and scripts.

The only API assumed necessary to read a Borg file, by a reasonably competent python, C, MATLAB or java programmer, is an HDF5 library (e.g., h5py in python, libhdf5 in C, JHI5 in java), or MATLAB. Other users, particularly those who are not strong programmers, will require an additional API. Borg provides facilities to store the general metadata that has been identified as important by the participants of the Neurodata Initiative. Borg's metadata representation is purposefully general, however, with reliance on human-readable text fields (such as copied from the Methods section of a paper) in lieu of attempting to define a general-purpose, forward-looking, machine-readable representation. This reduces the complexity of the initial file format version and makes implementation, adoption and specification easier/faster. After the format has been field tested and shown to work with software tools and to be usable in a lab environment, it is assumed the machine-

readability issue for metadata will be revisited (e.g., for version 2.0).

Top-level datasets

Top-level datasets are for file identification and version information.

Dataset	Type	Description	Comments
/neurodata_version	text	File version string	Eg, "Borg-1.0.0". This is the identifying string (e.g., "Borg") with trailing major, minor and patch numbers.
/identifier	text	Unique file ID	Eg, concatenated lab name, file creation date/time and experimentalist, or a hash of these and/or other values. The goal is that the string should be unique to all other files.
/file_create_data	text	Time file was created, UTC	Date + time, Use ISO format (eg, ISO 8601). File can be created after the experiment was run, so this may differ from experiment start time.
/session_start_time	text	Time of experiment/session start, UTC	Date + time, Use ISO format (eg, ISO 8601). All time units stored in file use this time as reference (ie, time zero)

TimeSeries

The file format is designed around a data structure called a *TimeSeries* which stores time-varying data. A *TimeSeries* is a superset of several INCF types, including signal events, image stacks and experimental events. To account for different storage requirements and different modalities, a *TimeSeries* is defined in a minimal form and it can be extended, or subclassed, to account for different modalities and data storage requirements. When a *TimeSeries* is extended, it means that the 'subclassed' instance maintains all of the data interface (eg, folders and datasets) as its parent and it has new folders and/or datasets of its own. Adding a new dataset/value pair should be equivalent to the Svoboda lab's key/value pair ("hash") approach. The *TimeSeries* makes this process of defining such pairs more hierarchical.

Each *TimeSeries* has its own HDF5 group/folder, and all datasets belonging to a *TimeSeries* are in that folder. The folder contains time and data components and users are free to add additional fields as necessary. There are two time objects represented. The first, *timestamps*, stores time information that is corrected to the experiment's time base (i.e., aligned to a master clock, with time-zero aligned to the starting time of the experiment). This field is used for data processing and subsequent scientific analysis. The second, ***sync***, is an optional folder that can be used to store the sample times as reported by the acquisition/stimulus hardware, before samples are converted to a common timebase and corrected relative to the master clock. This approach allows the Borg format to support streaming of data directly from hardware sources.

Dataset	**Type**	**Description**	**Comment**
data	polymorphic	Data values. Can also store binary data (eg, image frames)	This field may be a link to data stored in an external file, especially in the case of raw data.
timestamp	double array	Timestamps for the samples stored in *data*. Stores timestamps for every N'th data value.	Timestamps here have all been corrected to the common experiment master-clock. Time is stored as seconds and all timestamps are relative to experiment start time.
sync/	folder (optional)	Lab specific time and sync information as provided directly from hardware devices and that is necessary for aligning all acquired time information to a common timebase. The *timestamp* array stores time in the common timebase.	This folder will usually only be populated in *TimeSeries* that are stored external to the Borg file, in files storing raw data. Once *timestamp* data is calculated, the contents of 'sync' are mostly for archival purposes.

When data is streamed from experiment hardware, that data should be stored in a TimeSeries object, one *TimeSeries* per hardware stream, with each in its own HDF5 file. This allows the raw data files to be separate file-system objects that can be set as read-only once the experiment is complete. *TimeSeries* objects in /acquisition will link to the *data* field in the raw time series. In both /acquisition and /stimulus *TimeSeries*, the *timestamp* field will be populated by processed time&sync

data from the 'sync' folder(s).

The folder/group holding the *TimeSeries* has the following attributes:

description (text)	Description of the *TimeSeries*
source (text array)	Name of *TimeSeries* or *Modules* that serve as the source for the data contained here. It can also be the name of a device, for stimulus or acquisition data.
comments (text)	Human-readable comments about the *TimeSeries*. This second text field can be used to store additional information, or descriptive information if the primary description field is populated with a computer-readable string. If a required field of the time series is not populated (eg, if a quantitative value is not known, but it is known that the value was within a certain range), that information should be stored here.
ancestry (text array)	The class-hierarchy of this *TimeSeries*, with one entry in the array for each ancestor. An alternative and equivalent description is that this *TimeSeries* object contains the datasets defined for all of the *TimeSeries* classes listed. The class hierarchy is described more fully below. An example is: [0]="TimeSeries", [1]="ElectricalSeries" [1]="PatchClampSeries".
neurodata_type (text)	The string "TimeSeries"

The folder holding the *TimeSeries* can be used to store additional information (HDF5 datasets and attributes) beyond what is required by the specification. I.e., an end user is free to add additional key/value pairs as necessary for their needs. It should be noted that such lab-specific extensions may not be recognized by analysis tools/scripts existing outside the lab.

The *data* element in the *TimeSeries* will typically be an array of any valid HDF5 data type (e.g., a multi-dimentsional floating point array). The data stored can be in any unit. The attributes of the data field must indicate the SI unit that the data relates to (or appropriate counterpart, such as color-space) and the multiplier necessary to convert stored values to the specified SI unit. The data field has the following attributes:

si_unit (text)	The base unit for data stored in the *TimeSeries*. If no unit is appropriate, the most relevant identifier should be provided (e.g., which colorspace). SI units are preferred, with exception of temperature, as Centigrade is more commonly used than Kelvin.
conversion (float)	Scalar to multiply each element in *data* to convert it to the specified *si_unit*.
resolution (float)	Smallest meaningful difference between values in *data*, stored in units specified by *si_unit*.

The *timestamps* field has the following attributes:

rate (float)	Approximate data sampling or stimulus presentation rate (Hz), or exact rate, if known
t_interval (int)	Time stamps are stored for every N'th entry in data. This is N, and for now, N usually equals 1 (Karel's approach – store every timestamp). For cases where *rate* is used to calculate timestamps, *t_interval*=0 and only the first timestamp is stored (practical for Institute patch-clamp data). In a future version of the format, storing every Nth timestamp (Neuralynx-style) may be supported.

TimeSeries class hierarchy

The *TimeSeries* is a data structure/object. It can be "subclassed" to represent more narrowly focused modalities (e.g., electrical versus optical physiology) as well as new modalities (eg, video tracking of whisker positions). When it a *TimeSeries* is subclassed, new datasets can be added while all datasets of parent classes are preserved. An initial set of subclasses are described here. Users are free to define subclasses for their particular requirements.

AnnotationSeries extends TimeSeries

Stores, eg, user annotations made during an experiment. The TimeSeries::data[] field stores a text array, and timestamps are stored for each annotation (ie, t_interval=1). This is largely an alias to a standard TimeSeries but that is identifiable as annotation in a machine-readable way.

IntervalSeries extends TimeSeries

Stores intervals of data. The *timestamp* field stores the beginning and end of intervals. The *data* field stores whether the interval just started (>0 value) or ended (<0 value). Presently only +1 and -1 are being used, but it is possible to represent several interval types in the same series by using multiple key values (eg, 1 for feature A, 2 for feature B, 3 for feature C, etc). The field t_interval=1, and *data* stores an 8-bit integer. This is largely an alias to a standard TimeSeries but that is identifiable as being time intervals in a machine-readable way.

ElectricalSeries extends TimeSeries

Stores acquired voltage data from and extracellular recordings. The *data* field of an ElectricalSeries is an int or float array. Array structure: [num time samples] [num channels]. It contains all of the datasets of the basic *TimeSeries* as well as the following:

electrode_idx (int array)	Indices to electrodes described in the experiment's electrode map array (extracellular recordings under /general).

PatchClampSeries extends TimeSeries

Stores stimulus or response current or voltage. These are regular TimeSeries except for the addition of the following field:

electrode_name (text)	Name of electrode entry in /general/intracellular

VoltageStimulusSeries extends PatchClampSeries
CurrentStimulusSeries extends PatchClampSeries

These are aliases to standard PatchClampSeries. Their functionality is to better tag PatchClampSeries for machine (and human) readability of the file.

VoltageAcquisitionSeries extends PatchClampSeries

Stores data from intracellular voltage-clamp recordings. A *CurrentStimulusSeries* stores the stimulus injected. Each *VoltageClampSeries* stores information from a single current injection. The *VoltageClampSeries* has all of the datasets of an *ElectricalSeries* as well as the following:

capacitance_fast (float)	Units: Farads
capacitance_slow (float)	Units: Farads
resistance_comp_bandwidth (float)	Units: Hz
resistance_comp_correction (text)	Units: % (text because it's often reported as a range)
resistance_comp_prediction (text)	Units: % (text to keep consistent with correction)
whole_cell_capacitance_comp (float)	Units: Farads
whole_cell_series_resistance_comp (text)	Units: Ohms (text because it's often reported as a range)
gain (float)	Units: Volt/Amp

CurrentAcquisitionSeries extends PatchClampSeries

Stores data from intracellular voltage-clamp recordings. A *VoltageStimulusSeries* stores the stimulus injected. Each *CurrentClampSeries* stores information from a single current injection. The *CurrentClampSeries* has all of the datasets of an *ElectricalSeries* as well as the following:

bias_current (float)	Units: Amps
bridge_balance (float)	Units: Ohms. Was access resistance.
capacitance_compensation (float)	Units: Farads
resistance_compensation (float)	Units: Ohms
seal (float)	Units: Ohms
gain (float)	Units: Volt/Volt

SpikeEventSeries extends ElectricalSeries

Stores “snapshots” of spike events (i.e., threshold crossings) in *data*. This may also be raw data, as reported by ephys hardware. If so, the *TimeSeries::description* field should describing how events were detected. A *SpikeEventSeries* can also store derived event data that resides in a module. All events span the same recording channels and store snapshots of equal duration. *t_interval*=1 for this and all subclasses. Source of data should be provided in *TimeSeries::description*. *TimeSeries::data* array structure: [num events] [num samples] [num channels]. The *SpikeEventSeries* has all of the datasets

of an *ElectricalSeries*, as well as the following:

sample_length (int) Number of samples used to create a snapshot (eg, 32-samples). This is redundant with *data* but is explicit to detect if data is transposed versus what was expected.

ImageSeries extends TimeSeries

General image data that is common between acquisition and stimulus time series. Sometimes the image data is stored in the HDF5 file in a raw format while other times it will be stored as an external image file in the host file system. The *data* field will either be binary data or empty. The *ImageSeries* contains all of the datasets of the *TimeSeries* as well as the following:

format (text) Format of image. If this is "external" then the field *external_file* contains the path or URL information to that file. For tiff, png, jpg, etc, the binary representation of the image is stored in *data*. If the format is "raw" then the fields *bit_per_pixel* and *dimension* are used. For raw images, only a single channel is stored (eg, red)

external_file (text) Path or URL to external file, if *format* = "external".

bits_per_pixel (int) Only relevant for format='raw'. Bytes per image is int((bits_per_image+7)/8) – ie, no bit packing.

dimension (int array) Number of pixels on x, y and z axes. Only relevant for format='raw'. Data array may be byte-packed and use non-standard integer types (eg, int24), so we can't rely on HDF5's array dimensionality mechanism.

ImageIndexSeries extends TimeSeries

Stores indices to image frames stored in an ImageSeries. The purpose of the ImageIndexSeries is to allow a static image stack to be stored somewhere, and the images in the stack to be referenced out-of-order. This can be for the display of individual images, or of movie segments (as a movie is simply a series of images). The *data* field stores the index of the frame in the referenced ImageSeries, and the *timestamps* array indicates when that image was displayed. The ImageIndexSeries containes all datasets of *TimeSeries* plus the following:

image_stack (link) HDF5 link to ImageSeries containing images that are indexed

OpticalSeries extends ImageSeries

Image data that is presented or recorded. A stimulus template movie will be stored only as an image. When the image is presented as stimulus, additional data is required, such as field of view (eg, how much of the visual field the image covers, or how what is the area of the target being imaged). If the *OpticalSeries* represents acquired imaging data, orientation is also important. The *OpticalSeries* has all datasets of the *ImageSeries* as well as the following

field_of_view (float array) Width, height and depth of image, or imaged area (meters).

distance (float) Distance of camera/monitor from target/eye

orientation (text) Description of image relative to some reference frame (i.e., which way is 'up'). Must also specify frame of reference.

TwoPhotonSeries extends OpticalSeries

A special case of optical imaging. The *TwoPhotonSeries* has all the datasets of the *OpticalSeries* as well as the following:

max_voltage (float) (volts)

min_volrage (float) (volts)

pmt_gain (float) photomultiplier gain

wavelength (float) laser wavelength (nm or meters)

indicator (text) imaging indicator (e.g., GCaMP6s) and response wavelength

imaging_depth (float) Imaging depth into tissue, ie, below surface (meters).

scan_line_rate (float) Scan lines per second (Hz)

SpatialSeries extends TimeSeries

Direction, e.g., of gaze or travel, or position. The *TimeSeries::data* field is a 2-D array storing position or direction relative to some reference frame. Array structure: [num measurements] [num dimensions]. Each SpatialSeries has a text dataset *reference_frame* that indicates the zero-position, or the zero-axes for direction. For example, if representing gaze direction, "straight-ahead" might be a specific pixel on the monitor, or some other point in space. For position data, the 0,0 point might be the top-left corner of an enclosure, as viewed from the tracking camera. The units of *data* will indicate how to interpret *SpatialSeries* values. A SpatialSeries has all the datasets of a *TimeSeries* plus the following:

reference_frame (text) Description defining what exactly “straight-ahead” means

AbstractFeatureSeries extends TimeSeries

Abstract features, such as quantitative descriptions of sensory stimuli. The *TimeSeries::data* field is a 2-D array, storing those features (e.g., for visual grating stimulus this might be orientation, spatial frequency and contrast).

feature_description (text array) Description of the features represented in *TimeSeries::data*

Modules

Borg uses modules to store data for and represent the results of common data processing steps, such as spike sorting and image segmentation, that occur before scientific analysis of the data. Modules store the data used by software tools to calculate these intermediate results. Each module provides a list of the data it makes available, and it is free to provide whatever additional data that the module generates. Additional documentation is required for data that goes beyond standard definitions. All modules are stored in the /processing folder.

Like *TimeSeries*, *Modules* are each have their own folder where they store their data. Each module folder has the following attributes:

interfaces (text array) Names of the data interfaces offered by this module. For example, [0]=“EventDetection”, [1]= “Clusters”, [2]= “FeatureExtraction”. Interface descriptions are below.

source (text array) Names of the acquisition and stimulus *TimeSeries* directly used by the module as well as modules that this module has used data from (full paths – e.g., “/acquisition/timeseries/shank_2”).

module_description (text) Human-readable description of module and contents, for documentation and end-user information.

neurodata_type (text) The string “Module”

The *interface* list provides a machine-readable way to identify what information a module provides. Interfaces are not required to publish data and included interfaces can extend beyond what is listed here, so long as the data made available through the new interface is documented. To facilitate software identification of specific processing modules, the name of the module, preceded by the string “software:” (e.g., “software:kwik”) should also be in the *interfaces* list, even if all data published by the module is described by other interfaces (e.g., Clustering, FeatureExtraction, UnitTimes, etc).

The *source* field provides information that allows moving backward through the processing stream to find the source of data. Storage of additional provenance information (e.g., parameters, user decisions) is left to the module's developer, as the types of provenance information can vary greatly between modules.

Module Interfaces

Several module interfaces are listed. When a module provides an interface name in its *interfaces* list, it is required to have the data fields listed with that interface. An interface as described here is equivalent to Ken's 'contract'. The data published by each interface should reside in a folder in the module having the same name as the interface.

EventDetection

Detects spike events from voltage trace(s). EventDetection publishes the following datasets:

times (double array)

detection_method (text) Description of how events were detected, such as voltage threshold, or dV/dT threshold, as well as relevant values.

source_idx (int array) Indices into source *TimeSeries::data* array corresponding to time of event. Module description should define what is meant by time of event (e.g., .25msec before action potential peak, zero-crossing time, etc). The index points to each event from the raw data.

source_timeseries (link) *TimeSeries* that this data was calculated from. Metadata about electrodes and their

position can be read from that *TimeSeries* so it's not necessary to mandate that information be stored here.

EventWaveform

Represents either the waveforms of detected events, as extracted from a raw data trace in /acquisition, or the event waveforms that were stored during experiment acquisition. Event waveforms are stored as *SpikeEventSeries*. The name of each *SpikeEventSeries* is arbitrary but should be informative.

FeatureExtraction

Features, such as PC1 and PC2, that are extracted from signals stored in a SpikeEvent *TimeSeries* or other source. FeatureExtraction publishes the following datasets:

features (float array)	Multi-dimensional array of features extracted for each event. Array structure: [num events] [num channels] [num features]
times (double array)	Times of events that features correspond to (can be a link) Array structure: [num events]
description (text array)	Description of features (eg, "PC1") for each of the extracted features. Array structure: [num features]
electrode_idx (int array)	Indices to electrodes described in the experiment's electrode map array (under /general/extracellular_ephys). Array structure: [num channels]

FilteredEphys

Ephys data from one or more channels that has been subjected to filtering and downsampling. Examples of filtered data include Theta and Gamma (LFP has its own interface). FilteredEphys modules publish an *ElectricalSeries* for each filtered channel or set of channels. The name of each *ElectricalSeries* is arbitrary but should be informative. The source of the filtered data, whether this is from analysis of another time series or as acquired by hardware, should be noted in each's *TimeSeries::description* field. There is no assumed 1::1 correspondence between filtered ephys signals and electrodes, as a single signal can apply to many nearby electrodes, and one electrode may have filtered Theta and/or Gamma signals represented.

LFP

LFP data from one or more channels are published as an *ElectricalSeries*. The electrode map in the published *ElectricalSeries* will identify which channels are providing LFP data. Filter properties should be noted in the ElectricalSeries description or comments field.

BehavioralEvents
BehavioralEpochs
BehavioralTimeSeries

The objective of these interfaces is to provide generic hooks for software tools/scripts. This allows a tool/script to take the output one specific interface (e.g., UnitTimes) and plot that data relative to another modality data (e.g., behavioral events) without having to define all possible modalities in advance. Declaring one of these interfaces means that one or more *TimeSeries* of the specified type is published. These *TimeSeries* should reside in a folder having the same name as the interface. For example, if a BehavioralTimeSeries interface is declared, the module will have one or more *TimeSeries* defined in the module sub-folder "BehavioralTimeSeries". BehavioralEpochs should use *IntervalSeries*. BehavioralEvents is used for irregular events. BehavioralTimeSeries is for continuous data. Each *TimeSeries* of these generic interfaces should contain the following attribute:

interface_description (text)	Human-readable description of what the interface represents. E.g., "Right/left decision (right=1, left=0)".

Position

Position data, whether along the x, x/y or x/y/z axis. Each *Position* module publishes one or more *SpatialSeries*.

EyeTracking

Eye-tracking data, representing direction of gaze. Each *EyeTracking* module publishes one or more *SpatialSeries*.

PupilTracking

Eye-tracking data, representing pupil size. Each *PupilTracking* module publishes one or more *TimeSeries*.

CompassDirection

With a CompassDirection interface, a module publishes a *SpatialSeries* object representing a floating point value for theta. The *SpatialSeries::reference_frame* field should indicate what direction corresponds to "0" and which is the direction of rotation (this should be "clockwise"). The *si_unit* for the *SpatialSeries* should be "radians" with data stored on the interval [0, 2pi). Alternatively, "degrees" can be used.

UnitTimes

Event times in observed units (eg, cell, synapse, etc). The UnitTimes folder contains a dataset (double array) that is named after each unit. This double array stores the time, in seconds, of each unit event. These arrays should have a text attribute "source" that specifies where the event times originated from (e.g., a Clustering or Segmentation module). Name of array should match value in source module (e.g., name of ROIs from Segmentation module)

Clustering

Clustered spike data, whether from automatic clustering tools (e.g., klustakwik) or as a result of manual sorting. A Clustering module publishes the following datasets:

times (double array) Times of clustered events, in seconds. This may be a link to *event_times*. Array structure: [num events]

num (int array) Cluster number for each event Array structure: [num events]

source (FeatureExtraction module) Link to FeatureExtraction module that was the source of the clustered data. Can be link to self.

peak_over_rms (float array) Maximum ratio of waveform peak to RMS on any channel in the cluster (provides a basic clustering metric). Array structure: [num clusters]

ClusterWaveforms

The mean waveform shape, including standard deviation, of the different clusters. Ideally, the waveform analysis should be performed on data that is only low-pass filtered. This is a separate module because it is expected to require updating. For example, IMEC probes may require different storage requirements to store/display mean waveforms, requiring a new interface or an extension of this one. A ClusterWaveform module publishes the following datasets:

waveform_mean (2D float array) The mean waveform for each cluster, using the same indices for each wave as cluster numbers in the associated Clustering module (i.e, cluster 3 is in array slot [3]). Waveforms corresponding to gaps in cluster sequence should be empty (e.g., zero-filled).

waveform_sd (2D float array) Stdev of waveforms for each cluster, using the same indices as in mean.

waveform_filtering (text) Filtering applied to data before generating mean/sd

clusterwaveform_source (Clustering module) Link to Clustering module that was the source of the clustered data (can be link to this module, if it clustering information is here (i.e., it this module also has a Clustering interface)

ImageSegmentation

Stores list of pixels that represent region of interest (ROI) in imaging data. Pixel lists are here called masks, and masks are allowed to change with time (i.e., in case ROI is moving). All segmentation data is stored in a "segmentation" subfolder. Each ROI is stored in its own subfolder within s*egmentation*, with the ROI folder containing a list of successive masks and the timestamp that marks the start time for each mask (i.e., when the mask began to be used). Successive masks within each ROI folder should be named "mask_0", "mask_1", "mask_2", etc.
Also for masking neuropil.

ImageSegmentation/
- <image_plane>/ Folder name is human-readable description of imaging plane
  - attr: description (text) Description of image plane, recording wavelength, depth, etc
  - <roi_name>/
    - attr: description (text) Human-readable description of ROI

attr: id (text) Computer-readable, user-defined key for ROI
mask_0 (2D float array) Mask, represented in 2D ([x][y]) intensity image.
attr: timestamp (double) time when this mask starts to be used
attr: source (text) Path to ImageSeries that this mask applies to

Florescence

Florescence information about a region of interest (ROI). Storage hierarchy of florescence should be the same as for segmentation (ie, same names for ROIs and for image planes).

Florescense/
<image_plane>/ Folder name is human-readable description of imaging plane
attr: description (text) Description of image plane, recording wavelength, depth, etc
attr: id (text) Computer-readable, user-defined key for ROI
<roi_name> (*TimeSeries*) Florescense signal for ROI. TimeSeries name should match corresponding ROI name in the segmentation module.
roi_segments (text) New field in TimeSeries. Stores name of relevant ImageSegmentation module

DfOverF

dF/F information about a region of interest (ROI). Storage hierarchy of dF/F should be the same as for segmentation (ie, same names for ROIs and for image planes).

DfOverF/
<image_plane>/ Folder name is human-readable description of imaging plane
attr: description (text) Description of image plane, recording wavelength, depth, etc
attr: id (text) Computer-readable, user-defined key for ROI
<roi_name> (*TimeSeries*) dF/F signal for ROI. TimeSeries name should match corresponding ROI name in the segmentation module.
roi_segments (text) New field in TimeSeries. Stores name of relevant ImageSegmentation module

DrOverR TODO – Re-evaluate. Should be like DfOverF?

dR/R information, from intrinsic imaging.

dr_over_f (ImageSeries)

File organization, top-level groups (continued)

This is a more detailed description of the top level groups first presented on Page 1.

Top-level group: general

Experimental metadata, including animal strain, experimental protocols, experimenter, devices, etc, are stored under 'general'. While there are some very specific metadata fields defined here that are machine-readable, the general strategy used in the present Borg version is to store most metadata in free-form text fields, such as would appear in sentences or paragraphs from a Methods section. This is much faster and easer to implement, describe and use than a general-purpose machine-readable format.

Datasets & subgroups	Type	Description	Comments
session_id	text	Lab-specific ID for	Only 1 session_id per file, with all time aligned to experiment start time.
experimenter	text	Who performed the experiment	More than one person OK. Can specify roles of different people involved.
institution	text	Institution where experiment was performed	
lab	text	Lab where experiment was performed	

related_publications	text	Publication information.	PMID, DOI, URL, etc. If multiple, concatenate together and describe which is which.
notes	text	Notes about the experiment	
pharmacology	text	Description of drugs used, including how and when they were administered	Anesthesia(s), painkiller(s), etc., plus dosage, concentration, etc.
surgery	text	Narrative description of surgery/surgeries, including date(s) and who performed surgery	Much can be copied from Methods
protocol	text	Experimental protocol, if applicable	E.g., IACUC protocol
subject_id	text	ID of animal/person used/participating in experiment (lab convention)	
subject	text	Description of subject and where subject came from (eg, breeder, if animal)	
species	text	Species	
genotype	text	Genetic strain	If absent, assume WT
sex	text		
age	text	Age of person, animal, embryo	
weight	text	Weight at time of experiment, at time of surgery, and at other important times	
virus	folder	Information about viruses used in experiment	Underlying data stored in human-readable form, at least for now
virus_0	folder	Description of virus in free-form text, including virus ID, source, date made, injection location and volume, etc.	One of possibly many. Name is arbitrary but should be informative. Content can be from Methods section of paper.
slices	folder		
slice_0	text	Description of slice in free-form text, including information about preparation, thickness, orientation, temperature and bath solution.	One of possibly many. Name is arbitrary but should be informative. Content can be from Methods section of paper.
intracellular_ephys	folder		
filtering	text	Description of filtering used	Includes filtering type and parameters, frequency fall-off, etc. If this changes between *TimeSeries*, filter description should be stored as a text attribute for each *TimeSeries*.
electrode_0	folder	First of possibly many	Name should be informative. This can optionally be a free-form text field that includes relevant description and metadata.
description (attr)	text	Recording description, description of electrode (eg, whole-cell, sharp)	Free-form text (can be from Methods)
location	text	Area, layer, comments on estimation, stereotaxic coords (if in vivo), etc.	
device	text	Name of device in /general/devices	
slice	text	Name of entry in /general/slices, if relevant	
resistance	float	Electrode resistance	units: Ohm
extracellular_ephys	folder		
electrode_map	float array	Physical location of electrodes (x,y,z, in meters)	Location of electrodes relative to one another. This records the points in space. If an electrode is moved, it needs a new

			entry in the electrode map for its new location. Otherwise format doesn't support using the same electrode in a new location, or processing spikes pre/post drift.
electrode_group	int array	Identification number of probe, shank or tetrode each electrode resides on.	There's one entry here for each element in electrode_map. Integer in array maps to N in **probe_N**. All elements in an electrode group should have a functional association, for example all being on the same planar electrode array, oll on the same shank.
impedance	float array	Impedence of electrodes listed in electrode_map	
filtering	text	Description of filtering used	Includes filtering type and parameters, frequency fall-off, etc. If this changes between *TimeSeries*, filter description should be stored as a text attribute for each *TimeSeries*.
electrode_group_N	folder	One of possibly many folders, one for each electrode group.	"N" corresponds to value in electrode_group
location	text	Description of probe location	E.g., stereotaxic coords and other data, e.g., drive placement, angle and orientation and tetrode location in drive and tetrode depth
device	link	Name of device in /general/devices	
optophysiology	folder		
description (attr)	text	Virus description can optionally be stored as free-form text, including relevant description and metadata	This can be used with/instead-of the dedicated metadata fields (e.g., using an excerpt from Methods) to make use and adoption of the format easier.
setup	text	wavelength(s), indicators, equipment, etc	
location	text	stereotaxic coords of image center, plus orientation (ie, which way is "up")	
device	text	Name of device in /general/devices	
devices	folder	Description of hardware devices used during experiment.	Eg, monitors, ADC boards, microscopes, etc
device_0	text	One of possibly many. Information about device and device description.	Name should be informative. Contents can be from Methods.
machine_readable	folder	Content of folder mirrors revised version of computer readable metadata from 'what' document.	

Top-level group: acquisition

Acquired data includes tracking and experimental data streams (ie, everything measured from the system)

Datasets and subgroups	Type	Description	Comments
timeseries	folder	Acquired TimeSeries	When importing acquisition data to neurodata file, all acquisition/tracking/stimulus data must already be aligned to a common time frame. It is assumed that this task has already been performed.
timeseries_x	generic TimeSeries	One of possibly many acquisition TimeSeries objects.	Name is arbitrary. The TimeSeries::data[] dataset may be a link to raw acquired data in an external file.
images	folder		
image_x	binary	Photograph of experiment or experimental setup (video also OK)	Name is arbitrary

format (attr)	text	format of image	eg, jpg, png, mpeg
description (attr)	text	human description of image	If image is of slice data, include slice thickness and orientation, and reference to appropriate entry in /general/slices

If bulky data is stored in the /acquisition folder, the data can exist in a separate HDF5 file that is linked to by the file being used for processing and analysis.

Top-level group: stimulus

Stimuli are here defined as any signal that is pushed into the system as part of the experiment (eg, sound, video, voltage, etc). Many different experiments can use the same stimuli, and stimuli can be re-used during an experiment. The stimulus group is organized so that one version of template stimuli can be stored and these be used multiple times. These templates can exist in the present file or can be HDF5-linked to a remote library file.

Datasets and subgroups	Type	Description	Comments
templates	folder	Template stimuli	Time stamps in templates are based on stimulus design and are relative to the beginning of the stimulus. When templates are used, the stimulus instances must convert presentation times to the experiment's time reference frame.
timeseries_x	generic TimeSeries	One of possibly many templates, stored as a time series	Name is arbitrary.
presentation	folder	Stimuli presented during the experiment	
timeseries_x	generic TimeSeries	One of possibly many stimuli, stored as a time series	Name is arbitrary. The TimeSeries::data[] field for the stimulus may be a link to data in an external file.

Top-level group: epochs

An experiment can be separated into one or many logical intervals, with the order and duration of these intervals often definable before the experiment starts. In this document, and in the context Borg, these intervals are called 'epochs'. Epochs have acquisition and stimulus data associated with them, and different epochs can overlap. Examples of epochs are the time when a rat runs around an enclosure or maze as well as intervening sleep sessions; the presentation of a set of visual stimuli to a mouse running on a wheel; or the uninterrupted presentation of current to a patch-clamped cell. Epochs can be limited to the interval of a particular stimulus, or they can span multiple stimuli. The functionality of epochs is similar to some of that offered by NIX tags. It is possible to achieve some of NIX's multitag functionality. Different windows into the same time series can be achieved by including multiple instances of that time series, each with different start/stop times. Non-overlapping time series (e.g., all time series based on the same stimulus) can also be included in one epoch.

Datasets and subgroups	Type	Description	Comments
epoch_x		Different experimental epoch	Name is arbitrary but must be unique within the experiment.
neurodata_type (attr)	text	The string “Epoch”	
description	text		
start_time	double		
stop_time	double		
ignore_intervals	2D double array		How is this practically usable? Role of this field is to identify periods in an epoch that should be ignored (eg, periods of high noise/interference in an ephys recording). Can this be redesigned to be easier to use?
timeseries_x		One of possibly many input or output streams recorded during epoch	Name is arbitrary and does not have to match the TimeSeries that it refers to
idx_start	int	Epoch's start index in TimeSeries data[] field	This can be used to calculate location in TimeSeries timestamp[] field

count	int	Number of data samples available in this time series, during this epoch	This is used instead of stop_idx because of possible ambiguity of what 'stop_idx' means. Eg, if start_idx=3 and stop_idx=5, does this mean the interval { 3, 4 } or { 3, 4, 5 }
timeseries		link to TimeSeries	

Top-level group: processing

'Processing' refers to intermediate analysis of the acquired data to make it more amenable to scientific analysis. These are performed using *Modules*, as defined above. All modules reside in the processing folder. The modules residing here will hopefully be a superset of those defined by Ken.

Top-level group: analysis

The file can store lab-specific and custom data analysis without restriction on its form or schema, reducing data formatting restrictions on end users. Such data should be placed in the analysis group. The analysis data should be documented so that it is sharable with other labs.

Extending the format

Extensibility is handled by allowing users to add new datasets to existing TimeSeries or Modules, or defining new TimeSeries and/or Interfaces. All that's required is for a user/lab to provide documentation for those extensions when they share the file. There will be many extensions like this in the real world. When we find new TimeSeries or Interfaces that are popular, we can add these to the format specification. This approach allows extensibility without breaking backward compatibility.

Data integrity

HDF5 files can be come corrupted if a write operation fails, such as through a tool/script crashing or being manually closed, or by power failure. This is not unique to HDF5, as other programs (e.g., MATLAB) can produce corrupt files when they crash. A corrupted HDF5 file is effectively non-recoverable. Due to the nature and volume of data being stored in a neurodata file, Borg has data integrity requirements beyond that provided by HDF5 today.

HDF5 was the preferred neurodata format as indicated by many attendees of the first 'hackathon'. In order to use HDF5 and not an alternative corruption resistant file format (e.g., sqlite), Borg will use a multi-file storage mechanism for active files (i.e., files being actively used and written to during data processing and analysis). This approach will distribute many of the HDF5 groups from Borg into individual HDF5 files that are stored in a directory tree adjacent to the Borg file. For example, every module and module interface will reside in an independent HDF5 file. The contents of these files will be linked from the main Borg file, so when reading that file it will appear that everything resides in a single file, according to the above-described format. The write-API to Borg will manage storage in this directory tree. Distributed storage should be transparent to the user.

A detailed description of this directory storage approach will be produced at a later date. It should be possible to collapse all links to the secondary HDF5 files, producing a single Borg file. Secondary files should be duplicated before modify-write operations, allowing recovery in the event of a crash. In the event that the primary Borg file becomes corrupted, it should be reconstructible from the data stored in the directory tree.

Appendix VI

Excerpt of the NWB JSON specification (from: Neurodata Without Borders, 2016)


```
{
  "core": {
    "structures": {
      "extracellular_ephys/": {
        "<electrode_group_X>/": {
          "device": {
            "references": "/general/devices/devices/<device_X>/",
            "description": "Name of device(s) in /general/devices",
            "data_type": "text"
          },
          "location": {
            "description": "Description of probe location",
            "data_type": "text"
          },
          "description": {
            "description": "Description of probe or shank",
            "data_type": "text"
          }
```

```
            "_description": "One folder for each electrode group.  Name matches
group_id"
        },
        "impedance": {
            "description": "Impediance of electodes in electrode_map",
            "data_type": "text",
            "dimensions": [
                "electrode_number"
            ]
        },
        "filtering": {
            "description": "Description of filtering used.  If this changes between
TimeSeries, filter description should be stored as a text attribute for each TimeSeries.",
            "data_type": "text"
        },
        "electrode_group": {
            "references": "<electrode_group_X>/",
            "description": "Identification string for probe, shank or tetrode",
            "data_type": "text",
            "dimensions": [
                "electrode_number"
            ]
        },
```

```
"electrode_map": {
   "xyz": {
      "type": "struct",
      "components": [
         {
            "alias": "x",
            "unit": "meter"
         },
         {
            "alias": "y",
            "unit": "meter"
         },
         {
            "alias": "z",
            "unit": "meter"
         }
      ]
   },
   "description": "Physical location of electrode, x,y,z in meters",
   "data_type": "number",
   "dimensions": [
      "electrode_number",
      "xyz"
```

```
            ]
        }
    },
    "protocol": {
        "description": "Experimetnal protocol, if applicable (e.g., IACUC)",
        "data_type": "text"
    },
    "slices": {
        "description": "Description of slices, including information about preparation thickness, orientation, temperature and bath solution",
        "data_type": "text"
    },
    "<TimeSeries>/": {
        "merge": [
            "<timestamps>/"
        ],
        "data": {
            "dimensions": [
                "timeIndex"
            ],
            "data_type": "number"
        }
```

Appendix VII

The 'What' Document

Data types

Timeseries: a numeric array with time in seconds on a dimension scale
Events: a numeric array of times in seconds
Epochs: a numeric array of start and end times in seconds

Module 0: experiment and animal information

general
Experimenters – Text; can be more then one person
SourceLab – Text; labhead name
Institution – Text; place where the experiment was performed
ExperimentDate – Text # international standard notation
TimeOfExperimentStart – Text # international standard notation
ReverseCycle -
DOI -
PMID -
relatedPublication - # Text or URL

Animal
SubjectID – Text (lab convention, provided each animal is uniquely identified)
Species – Text (use biology-wide standard)
Genotype - Text (use biology-wide standard) – Rob follow up to look at standards
Source - Text (use data standard)
Sex - Text (M/F)
Age - Text (days) # at start of experiment
Weight before experiment – float (kg)
Weight after experiment float (kg)

These next fields can be copied from the methods section
Surgery - text field with narrative description
Date of first surgery – text # at start of experiment
Anesthesia – text description (details quantified later)

virus details – machine readable
For v in Viruses (one entry for each distinct type, not each injection)
    VirusID – upenn ID or equivalent
    Virus source
    Virus lot
    Date made
    nominal titer

for i in Injections (one entry for each injection per type)
stereotaxic_location (millimeters) – numeric # (AP, ML, depth)
location description – text field # includes how location was determined; e.g ARA
injection volume – numeric (nanoliters)
rate of injection – numeric (nanoliter / minute)
injection date – text

slicing details (if appropriate)
Brain slice preparation – text
temperature – scalar or timeseries (Celsius)
bath solution – text, copy from methods paper

Module 1: Intracellular electrophysiology

Type: text (two options: "In vitro" or "In vivo")
a text field with predefined options ensures machine and human readability

For e in Electrodes
Stereotaxic location - numeric (if in vivo)
Location description – text (e.g. area, layer, comments on how estimated)
Allen location ontology

#amplifier info
Equipment - text
gain – numeric (in samples/ V?)
frequency roll-off – numeric (Hz)
capacitance compensation – text
resistance compensation – text
Electrode properties – text ("whole cell", "sharp", etc…)

if the electrode was used for current clamp, the next entries will be there
Vm - time-series
Icommand – time-series

if the electrode was used for voltage clamp, the next entries will be there
Im – time-series
Vcommand – time-series

note: units for timeseries are stored as HDF5 attributes

Comments:

Pharmacology will be specified in a separate pharmacology module Anything can be commented or extended in private fields

Module 2: Extracellular electrophysiology

note this document only describes public data: data describing the spike sorting provenance will be stored in additional submodules, in a format designed by the algorithm designer

Recording type – text (options: “chronic” or “acute”)

information about electrodes
note this is organized by channel group (i.e. shank) rather than by physical probe
For c in channel groups
    probe source – text (manufacturer, part number)
    probe ID – integer (says which physical probe this shank belonged to)
    shank geometry for all sites – numerical array, relative to shank tip
    location of shank tip – stereotactic coordinates
    location of shank tip histology – free text
    reference – text
    ground – text

LFP – numeric array (for all channels)

For M in Units
    # computed data
    spike train – event series
    channel group of origin – integer (ref to channel groups structure)
    mean spike waveform – numeric (N_samples x N_channels_in_group)

    # Derived information about unit
    Cluster type – text (“SUA”, “MUA, “noise”): human decision
    Spike width – numeric (seconds)
    Estimated cell location – derived quantity (e.g. center of mass of waveform)

    # the next fields are examples of ways to give information on spike quality. Array names will specify which metrics are used; more can be added
    Isolation distance – numeric
    L_ratio – numeric
    Refractory violation rate - numeric

    # the next fields are examples of multiple ways to identify neuronal classes. Array names will specify which criteria are used
    Spike width class – text (“narrow”, “wide”)

Optogenetic response - binary

Comment:
How do we link units that are the same across sessions
Provenance of how clustering was done is stored in a submodule that depends on the spike sorting algorithm. The spike sorting designer has to provide this information.

Module 3
Optophysiology – raster scanning

general information about experiment

experiment metadata (*: required)
*excitation_lambda – numeric (wavelength in nm)
*emission_lambda1 – centerwavelength & spectral width, 'green'
emission_lambda2 – centerwavelength & spectral width, 'red'
*imaging indicator – text (one of a few allowed strings e.g, "GCaMP6s") - # helps with facility of use
imaging rate – numeric (e.g 7 Hz) % implicit in dt
*reference_coord – format TBD (in um), but all coordinates are in reference to this (may need to be 6DoF to denote orientation) # One reference point
optical_axes – two angles to specify if different from orientation of reference_coord
reference_coord_data – stores information related to reference coordinates - ISI, post-hoc histology of barrels, etc.

data pertinent to full experiment (*: required)
*plane_size – horizontal, vertical plane size in distance units (um; units)
*plane_size_pixels – horizontal, vertical pixel count
*n_subvolumes – number of subvolumes # typically 1; if tiling etc more
*n_planes – how many planes per subvolume? # 1 for single plane
*plane_image – size plane_size_pixels, one per plane # image in which ROIs are drawn
*plane_coord – maps plane_image to reference_coord (e.g. top-left corner)

reference_stack – high-resolution image stack(s), can be multichannel # in case there is separate high-res data; posthoc, for example
reference_stack_transform – 3-fold repeat of reference-stack, where one of each maps X Y and Z of each pixel in reference_stack to reference_coord (0,0,0) in units of pixel
plane_reference_stack_transform – for each plane 3 integral matrices same size as image (X, Y, Z), maps plane into the reference_stack pixel space with 0,0,0 being the top-left of the first image in reference_stack

link to source data – this needs to be incorporated in *all* modules
*raw_data_file_list – list of raw data files (.tif, etc.)

*raw_data_map – for each plane, a mapping of same length as the time vector for rois beloging to this plane and having 2 rows, where row 1 indicates which raw data file this datapoint came from, and row 2 indicating the frame within the file

for r in ROIs

*id – unique identifier for this ROI

*source_plane – which plane roi is part of; the definition of a source_plane is that all ROIs with that source plane have the same time base

*roi_type – this should be a flexible list with an agreed "tag" vocabulary (e.g., soma, nucleus, multi-cell, axon, axon group, etc.; things like VIP-positive, GFP-positive, retrograde labeled; and things like ablated, ChR2+ expressing)

*roi_mask – image, binary or float

*f – time series showing fluorescence of roi over experiment

fo – time series; resting fluorescence or reference fluorescence channel

for neuropil, you have a second ROI with roi_type "neuropil"

Module 4: Sensory stimuli

For R in stimulation devices

specification of device – text (e.g. video; loud speaker; tactile stimulus)

for S in distinct stimuli (for each stimulator)

description of stimulus (polymorphic e,g – link to a file; numbers specifying grating orientation / contrast spatial frequency)

for P in presentations (i.e. repeats of stimulus S)

epoch

Module 5: Simple optogenetic stimuli

for D in stimulation devices

stimulation device description – text (e.g. laser model, LED)

lambda – scalar (wavelength in nm)

Stereotaxic location - numeric

Location description – text (e.g. area, layer, comments on how estimated)

Allen location ontology

For S in stimuli (one for every time the laser was turned on)

epoch

waveform – time series

mean power – scalar (which unit? Annotation: measured or nominal)

Comment:
This is supposed to deal with 'fibers' and LEDs. More complex schemes, such as laser scanning and projection systems (e.g. holography) will have separate modules.

Module 6: Behavioral events

For T in event types
    description of event type – text (e.g. “lick left”; “trial start”; “touch”)
    description of event – text (including device)
    list of data fields; description of data fields; units
    for E in events
        time – numeric (seconds)
        additional data (e.g. exact tongue location; strength of touch)

for P in epoch types
    description of epoch type (e.g. lever press; whisker touch)
    description of epoch – text (including device)
    list of data fields; description of data fields; units
    for E in epochs
        time start, time end
        addition data (e.g. timeseries giving lever force)

Module 7: Pharmacology during the experiment

For d in drugs
    identity – text
    administration route – (bath; IV, IP, IM etc)
    for a in administrations
        epoch – start time equals end time for single injection
        volume – if applicable
        concentration – numeric, units (g/kg; mM; %)

References

about-alm-1. (n.d.), Retrieved 2015 from https://crcns.org/data-sets/motor-cortex/alm-1/about-alm-1

about-ret-1. (n.d.), Retrieved 2015 from https://crcns.org/data-sets/retina/ret-1/about-ret-1

about-ssc-1. (n.d.), Retrieved 2015 from https://crcns.org/data-sets/ssc/ssc-1/about-ssc-1

allen_inst_ephys_summary.pdf (n.d.), Retrieved 2015, from https://crcns.org/files/data/pvc-6/allen_inst_ephys_summary.pdf

A Standard for Neuroscience Data. (n.d.). Retrieved 2015, from http://www.crd.lbl.gov/news-and-publications/news/2014/a-standard-for-neuroscience-data

Brain Format. (n.d.). Retrieved 2015, from https://bitbucket.org/oruebel/brainformat

BRAIN Initiative. (n.d.). Retrieved 2015, from http://www.whitehouse.gov/share/brain-initiative

Breaking Down the Data Barriers in Neuroscience. (n.d.). Retrieved 2014, from http://www.kavlifoundation.org/science-spotlights/breaking-down-data-barriers-neuroscience#.VEx7ddR4q0c

crcns_alm-1_data_description.pdf. (n.d.). Retrieved 2015 form https://crcns.org/files/data/alm-1/crcns_alm-1_data_description.pdf

crcns-hc3-data-description.pdf. (n.d.). Retrieved 2015 from https://crcns.org/files/data/hc3/crcns-hc3-data-description.pdf

crcns-hc3-processing-flowchart.pdf. (n.d.). Retrieved 2015 from https://crcns.org/files/data/hc3/crcns-hc3-processing-flowchart.pdf

crcns_ret-1_data_description.pdf. (n.d.). Retrieved 2015 from https://crcns.org/files/data/ret-1/crcns_ret-1_data_description.pdf

crcns_ssc-1_data_description.pdf. (n.d.). Retrieved 2015 from https://crcns.org/files/data/ssc-1/crcns_ssc-1_data_description.pdf

Data Sets. (n.d.). Retrieved 2015, from http://crcns.org/data-sets

Diba, K., & Buzsáki, G. (2008). Hippocampal network dynamics constrain the time lag between pyramidal cells across modified environments. The Journal of Neuroscience, 28(50), 13448-13456.

Fletcher, M., Liang, B., Smith, L., Knowles, A., Jackson, T., Jessop, M., & Austin, J. (2008). Neural network based pattern matching and spike detection tools and services—in the CARMEN neuroinformatics project. Neural Networks,21(8), 1076.

Garcia, S., Guarino, D., Jaillet, F., Jennings, T., Pröpper, R., Rautenberg, P. L., Rodgers, C. C., Sobolev A., Wachtler T., Yger P.,Davison, A. P. (2014). Neo: an object model for handling electrophysiology data in multiple formats. Frontiers in Neuroinformatics, 8, 10.

Gardner, D. (2004). Neurodatabase. org: networking the microelectrode. Nature neuroscience, 7(5), 486-487

Gibson, F., Overton, P., Smulders, T., Schultz, S., Eglen, S., Ingram, C., Panzeri, S., Bream, P., Whittington, M., Sernagor, E., Cunningham, M., Adams, C., Echtermeyer, C., Simonotto, J., Kaiser, M., Swan, D., Fletcher, M., and Lord, P. (2009) Minimum Information about a Neuroscience Investigation (MINI): Electrophysiology.

G-Node/nix. (n.d.). Retrieved 2015, from https://github.com/G-Node/nix

Grewe J, Wachtler T and Benda J (2010). odML Format and Terminologies for Automated Handling of (Meta)data. Front. Neurosci. Conference Abstract: Neuroinformatics 2010

Grewe J, Wachtler T and Benda J (2011) A bottom-up approach to data annotation in neurophysiology. Front. Neuroinform. 5:16.

Guo, Z. V., Li, N., Huber, D., Ophir, E., Gutnisky, D., Ting, J. T., ... & Svoboda, K. (2014). Flow of cortical activity underlying a tactile decision in mice. Neuron, 81(1), 179-194.

HDF Group - HDF5 - Hierarchical Data Format. (n.d.), Retrieved 2014 , from http://www.hdfgroup.org/HDF5/

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. The Journal of physiology, 117(4), 500.

INCF Neuroinformatics Portal. (n.d.). Retrieved 2015, from http://www.incf.org/

International Neuroinformatics Coordinating Facility. (n.d.). Retrieved 2015, from http://en.wikipedia.org/wiki/International_Neuroinformatics_Coordinating_Facility

Kemp B, Värri A, Rosa AC, Nielsen KD & Gade J. (1992). A simple format for exchange of digitized polygraphic recordings.

Kemp B., Olivan J. (2003). European data format 'plus' (EDF+), an EDF alike standard format for the exchange of physiological data. Clin. Neurophysiol. 114, 1755

Klusta-team/kwiklib. (n.d.), Retrieved 2015, from https://github.com/klusta-team/kwiklib/wiki/Kwik-format

Lefebvre, J. L., Zhang, Y., Meister, M., Wang, X., & Sanes, J. R. (2008). γ-Protocadherins regulate neuronal survival but are dispensable for circuit formation in retina. Development, 135(24), 4141-4151.

Le Franc Y, Bandrowski A, Brůha P, Papež V, Grewe J, Mouček R, Tripathy SJ and Wachtler T (2014). Describing neurophysiology data and metadata with OEN, the Ontology for Experimental Neurophysiology. Front. Neuroinform. Conference Abstract: Neuroinformatics 2014

Li, N., Chen, T. W., Guo, Z. V., Gerfen, C. R., & Svoboda, K. (2015). A motor cortex circuit for motor planning and movement. Nature, 519(7541), 51-56.

Mizuseki, K., Sirota, A., Pastalkova, E., & Buzsáki, G. (2009). Theta oscillations provide temporal windows for local circuit computation in the entorhinal-hippocampal loop. Neuron, 64(2), 267-280.

Multiscale Electrophysiology File Format. (n.d.). Retrieved 2014, from https://www.ieeg.org/sites/default/files/MEF_Format.pdf

Neo - NeuralEnsemble. (n.d.). Retrieved 2015, from http://neuralensemble.org/neo/

Neurodatabase. (n.d.). Retrieved January 30, 2015, from http://neurodatabase.org/

Neurodata Without Borders. (n.d.). Retrieved 2016, from https://neurodatawithoutborders.github.io/

Neurodata Without Borders | The Kavli Foundation. (n.d.). Retrieved 2016, from http://www.NWB.org/

Neurodata Without Borders meeting report. (n.d.). Retrieved 2015, from https://incf.org/activities/projects/neurodata-without-borders-meeting-report

NWBh1_09_Keith_Godfrey.pdf. (n.d.). Retrieved 2015, from https://crcns.org/files/data/NWB/h1/NWBh1_09_Keith_Godfrey.pdf

NWB_hackathon1.pdf (n.d.). Retrieved 2015, from https://crcns.org/files/data/NWB/NWB_hackathon1.pdf

NeXus Scientific Data Format. (n.d.). Retrieved 2016, from http://www.nexusformat.org/

Peron, S. P., Freeman, J., Iyer, V., Guo, C., & Svoboda, K. (2015). A cellular resolution map of barrel cortex activity during tactile behavior. Neuron, 86(3), 783-799.

Rautenberg, P., Sobolev, A., Herz, A. V. and Wachtler. T. , A Database System for Electrophysiological Data. Transactions on Large-Scale Data- and Knowledge-Centered Systems IV. Ed., Berlin Heidelberg: Springer, 2011. 1-14. Print.

Reardon, S. (2014), Brain-mapping projects to join forces. Nature, 18

Recent News. (n.d.). Retrieved 2014, from http://www.kavlifoundation.org/kavli-news/prominent-us-research-institutions-announce-collaboration-toward-sharing-and#.VDgBBNTF8YG

Science Funders Hope to Link Up Large Scale Brain Research. (n.d.). Retrieved 2015, from http://www.insidephilanthropy.com/home/2014/8/8/science-funders-hope-to-link-up-large-scale-brain-research.html

Sobolev, A., Stoewer, A., Pereira, M., Kellner, C. J., Garbers, C., Rautenberg, P. L., & Wachtler, T. (2014). Data management routines for reproducible research using the G-Node Python Client library. Frontiers in Neuroinformatics,8, 15.

Sobolev, A., Stoewer, A., Leonhardt, A., Rautenberg, P. L., Kellner, C. J., Garbers, C., & Wachtler, T. (2014). Integrated platform and API for electrophysiological data. Frontiers in Neuroinformatics, 8, 32.

Schlögl A. (2006). GDF – a general dataformat for biosignals. Comput. Res. Repository. arXiv:cs/0608052v10.

Schlögl A. (2010). An overview on data formats for biomedical signals, in World Congress on Medical Physics and Biomedical Engineering, Vol. 25/4, eds Dssel O., Schlegel W., editors. (Munich; Berlin; Heidelberg: Springer;), 1557–1560

Schwarz, D. A., Lebedev, M. A., Hanson, T. L., Dimitrov, D. F., Lehew, G., Meloy, J., ... & Nicolelis, M. A. (2014). Chronic, wireless recordings of large-scale brain activity in freely moving rhesus monkeys. Nature methods, 11, 6.

Svoboda_lab_data_format_general.pdf. (n.d.). Retrieved 2015, from https://crcns.org/files/data/alm-1/Svoboda_lab_data_format_general.pdf

Swagger. (n.d.). Retrieved 2016, from http://swagger.io/

Teeters, J. L., Godfrey, K., Young, R., Dang, C., Friedsam, C., Wark, B., et al. (2015). Neurodata Without Borders: Creating a Common Data Format for Neurophysiology. Neuron, 88(4), 629-634.

Teeters J. L., Benda J., Davison A. P., Eglen S., Gerhard S., Gerkin R. C., et al. (2013). Considerations for developing a standard for storing electrophysiology data in HDF5. Front. Neuroinform. Conference Abstract: Neuroinformatics 2013, 69.

Welcome to CARMEN. (n.d.). Retrieved 2015, from http://www.carmen.org.uk/

Welcome to the CRCNS data sharing website. (n.d.). Retrieved 2015, from https://crcns.org/

Welcome to the German Neuroinformatics Node. (n.d.). Retrieved 2015, from http://www.g-node.org/

Zehl L, Denker M, Stoewer A, Jaillet F, Brochier T, Riehle A, Wachtler T and Grün S (2014). Handling complex metadata in neurophysiological experiments. Front. Neuroinform. Conference Abstract: Neuroinformatics 2014.